ITU A|Z
VOL: 11, NO:1 138-157, 2014-1

Design games: A conceptual framework for dynamic evolutionary design

N. Onur SÖNMEZ*, Arzu ERDEM**
**Istanbul Technical University, Faculty of Architecture, Istanbul, TURKEY & Delft University of Technology, Faculty of Architecture and the Built Environment, Delft, the NETHERLANDS*
***Istanbul Technical University, Faculty of Architecture, Istanbul, TURKEY*

Received: May 2012 *Final Acceptance: March 2014*

Abstract:
Most evolutionary computation (EC) applications in design fields either assume simplified, static, performance-oriented procedures for design or focus on well-defined sub-problems, to be able to impose problem-solving and optimization schemes on design tasks, which render known EC techniques directly applicable. However, in most design situations, well-defined and static problems are not given, but must be constructed from messy situations, and the definition of a problem takes place during the solution process. Thus, evolutionary design requires contextual and dynamic problem definition and evaluation procedures, which has not yet been realized through EC. This study sets out for a critical reappraisal of EC for design, and proposes a conceptual framework as a research tool for the exploration of dynamic evolutionary design. After a critical review of EC in design, the article discusses its claims with reference to design theory, outlines the framework, and examines dynamic evolutionary strategies and required intelligent technologies. Although tackling a practical task, or solving the problem of dynamic evolution are not aimed in this study, an experimental application based on the framework will be presented in detail, to exemplify a mapping between the rather abstract concepts of the framework and the operators of a specific evolutionary algorithm.

Keywords: *Evolutionary design, AI in design, design automation, evolutionary computation.*

1. Introduction

Evolutionary computation (EC) denotes a family of techniques within computer science, which are inspired by the processes of biological evolution. EC typically exploits mechanisms like variation, reproduction, and selection and is often used for problem-solving and optimization, especially when the problems do not lend themselves to easily applicable algorithmic procedures. There are many different variants of evolutionary algorithms (EAs), which are united by a common underlying idea: “given a population of individuals, the environmental pressure causes natural selection (survival of the fittest), which causes a rise in the fitness of the population” (Eiben and Smith, 2003, p. 15). The most dominant application area of EC has been optimization problems (de Jong, 206, p. 23), where EC has been considered

as a problem-solving technique, which tries to approach optimal values closer and closer through the migration of a species of solution candidates within a complex search space (de Jong, 2006, p. 71). The use of EC within design fields has also mostly followed problem-solving and optimization paths. However, such approaches, which are suitable for engineering problems, are not sufficient for understanding design situations or carrying out design tasks. Through a critical review of existing studies and with reference to design theory, we will claim below that there is a need for a reappraisal of EC in design and will state the rationale for the proposal of a new framework that resides on a general level.

There has been a wide range of attempts to utilize EC for design and arts. For instance, EC has been utilized, mostly experimentally, for product design (Liu and Tang, 2006; Ang et al., 2006) and for generating two-dimensional forms and graphic layouts (Geigel and Loui, 2001; Ross, Ralph, and Zong, 2006; da Silva Garza, Lores, and Zamora, 2008). EC offers mostly generic mechanisms in the form of specific EAs, and problem definition and evaluation approaches are the core mechanisms that enable the adaptation of an EA to a specific task. In the above studies, EC was utilized for parts of the overall design task and only after the strict definition of suitable problems, which is congruent with the optimization model.

EC has been very popular in generative art circles, at least since, in his book "The Blind Watchmaker" Richard Dawkins described how evolution could be used to evolve shapes (Dawkins, 1996, pp. 43-74; Lewis, 2008). Following Dawkins, and Genetic Programming, Karl Sims and William Latham[1] evolved two-dimensional abstract illustrations in the early 1990s (Lewis, 2008). These studies were using tree-based mathematical expressions for genotype representation, which has been adopted in most subsequent evolutionary art. Most of these studies depend upon the same, pre-given way of painting digital canvasses, i.e., a fixed problem definition. Nevertheless, the often-assumed interactive evaluation approach removes the need for static fitness definitions and gives such studies a proportion of dynamism. However, interactive evolution is tiresome and does not fit well with the aims towards automation. Karl Sims has also experimented with artificial life[2], which subsequently became a research area on its own (Lewis, 2008). Artificial life may assume a co-adaptation approach much like the aimless, open-ended natural evolution. This potentially removes fixed problem definitions and brings EC outside the bounds of problem-solving approaches, yet it is not straightforward to utilize the adaptation model for design situations, which involve goal directed tasks.

John Frazer (1995) has been a precursor of the usage of EC in architecture. John Gero's research group studied a diverse array of tasks through EC. They experimented with space layout topologies, combination of shape grammars with evolutionary approaches, and evolving linear plan units as design genes within a two-phased hierarchical evolution (Gero, Louis, and Kundu, 1994; Gero, Schnier, and Thorsten, 1995; Damski and Gero, 1997; Gero and Kazakov, 1998; Jo and Gero, 1998). Rosenman (1997) studied interactive evolution for floor plan generation. Rosenman and Saunders (2003) experimented with a self-regulatory hierarchical co-evolution model for designing. These studies operated within highly constrained, simplified, and isolated sub-domains of architecture and mostly followed optimization approaches using just a few objectives, such as circulation costs calculated through pre-given adjacency matrices. Given the direct borrowing of EC from

[1] Artist's website available at [online]: <http://www.doc.gold.ac.uk/~mas01whl/> (Accessed: February 2014).

[2] For artificial creatures evolved by Sims [online]: <http://archive.org/details/sims_evolved_virtual_creatures_1994> (Accessed: May, 2013).

applied engineering fields, which imposed the problem-solving and optimization approaches, it is not surprising that the more developed applications appeared within rather well-defined sub-problems of architecture. A series of experimentations has been carried out by Caldas, Norford, and Rocha (Caldas and Rocha 2001; Caldas and Norford, 2002, 2003; Caldas, 2003, 2005, 2006, 2008), which uses EC for the aim of integral building envelope design and performance optimization. Again, with a performance oriented design approach, Turrin, Buelow, and Stouffs developed an application to combine parametric modeling and EC (Turrin et al., 2011; Turrin, von Buelow, and Stouffs, 2011). These studies went on to assume a simplified, performance-oriented procedure for design, which rendered known EC techniques directly applicable.

Using EC for the optimization phases of design is rather straightforward; however, what most of the above listed studies do is to implicitly carry out a reformulation of design to make it compatible with the problem-solving model. This did not seem problematical to most researchers, because the problem-solving paradigm, which, sought out to understand design as a rational problem-solving process in order to enable the handling of design tasks through search and optimization methods (Simon, 1996), has been a dominant influence shaping prescriptive and descriptive design methodology, and a considerable portion of the work done in design methodology has followed it in its assumptions, view of science, goals, and methods (Dorst and Dijkhuis, 1995). In the problem-solving paradigm, the design process can be said to comprise three major tasks, which echoes the tripartite phase models of design process (Kalay, 1992): 1) Defining a set of desired conditions that comprise the objectives to be achieved, i.e., analysis (posing goals, objectives, performance criteria, constraints, etc.), 2) Specifying actions that will achieve the desired objectives, i.e., synthesis / generation (search operators, modifiers, etc.), and 3) Predicting and evaluating the effects of the specified actions to verify that they are consistent with each other and they achieve the desired objectives, i.e., evaluation (simulation, testing, etc.).

Indeed, the 'pose-search-evaluate-choose' process can be recognized in real design processes, although in a rather chaotic manner (Lawson, 2005, p. 49). However, it would not be wise to think that all design activity can be captured within the problem-solving model (Lawson, 2005, p. 31). In contrast to well-defined (alternatively: well-formulated, well-structured) problems, the problems that the designers tackle often exhibit characteristics that are referred to as ill-defined, ill-structured (Simon, 1973), open-ended and even as wicked (Rittel and Webber, 1973). In most design situations, well-defined problems are not given but must be constructed from messy problematic situations (Schön, 1983, p. 47), and the primary task is not the optimization procedure, but the definition of the problem together with the solutions.

Every design situation involves both open and closed types of problems. These two types of problems require different mental strategies (Dorst, 2006, p. 15, 16, 35; Cross, 2006, p. 77). The intensity and essentiality of the open problems in a design situation would render its character as open or closed. Closed design problems are more like puzzles. Solutions are tried out, and the feedback from evaluating the solutions is immediate and clear. On the other hand, open problems require the designer to play with concepts and ideas through a wide range of possibilities before settling on a firm direction. Ideas are proposed, critically inspected, and continuously

reconsidered. This is done repeatedly, making gradual improvements as the designer learns more about the problem. Therefore, design involves finding appropriate problems, as well as solving them, and includes substantial activity in problem structuring and formulating, rather than merely accepting the problem as given. A cognitive neuroimaging experiment by Alexiou, Zamenopoulos, and Johnson (2009) supports the idea that these two types of problems, i.e., open / ill-defined and closed / well-defined types, which were equated to design and problem-solving respectively, require different mental strategies.

Design often begins without any clear statement of the problem as a whole. Some general objectives may exist, but there is rarely an unambiguous way of knowing how well one is doing as one proceeds. Even more confusingly, it might not be possible to arrive at an overall assessment to compare relative values of various solutions. Design solutions are not certainly right or wrong and there are no knowable optima (Dorst and Dijkhuis, 1995; Lawson, 2004, p. 20). For these reasons, at least until its detailing stages, a design process cannot be identical to optimization. The optimization approach assumes that the essential needs of a circumstance can be listed and can be expressed in a measurable form before the solution process starts. For design problems, even if this was possible, the quantities of criteria, constraints, or needs could easily reach huge numbers. Moreover, these requirements mostly reside on incommensurate levels, so that comparing them is also a problem itself. After all, designers are rarely completely sure of these needs, let alone being able to formulate them in a measurable form.

Several architects have been interested in EC as a generative tool and developed specific, 'one-off' design approaches (Chouchoulas, 2003; Chouchoulas and Day, 2007; Hemberg et al., 2008; Dillenburger et al., 2009). However, it is not easy to adapt such highly specific approaches to other design situations, which is important in building generic design tools or methods. In all the above studies, as well as in the 'one-off' approaches, EC has been used through static problem definitions, which do not respond well to the essentially vague, highly contextual, and consequently, highly dynamic manner of design processes. A second problem arises due to simplified evaluations, which come nowhere near a real designer's subtlety in evaluating a solution alternative. Multi-objective evolution is problematic for tasks that require complicated representations and a high amount of conflicting objectives, which is mostly the case with design. The only way forward appears as devising dynamic evolutionary processes, where at any juncture a limited number of operations and evaluations would be applied. However, this requires a contextual intelligence capable of appreciating which operator to use and how, throughout the changing design context.

Therefore, evolutionary design has yet to find the paths that would account for the dynamic aspects of design, which necessitates an additional layer of research between design theory and specific and pragmatic applications. More specifically, there is a requirement for new conceptualizations and frameworks that target the use of EC within ill-defined situations, such as the Situated FBS Ontology (Gero and Kannengeisser, 2004, 2007) and Janssen's framework for evolutionary architectural design (Janssen, 2004, 2006).

We propose the two main requirements of evolutionary design as, 1) contextual, flexible, and dynamic (on the run) problem definition, and 2)

more refined, multi-faceted, and again contextual and dynamic evaluation. The technology required for these aims is yet to be developed or adopted from other fields. Nevertheless, there are already clues for potential tracks. Sean Hanna (2005, 2006, 2007) utilized machine-learning techniques to generate implicit objective functions for usage in evolutionary generation of floor layouts. Within an on-line monitoring mechanism, such an approach may enable contextual evaluation and dynamic changes within an evolutionary system.

In brief, this study attempts at developing an intermediary evolutionary design framework, which has to be flexible, abstract, and open-ended, not only because it has to be applied in varying situations and tasks, but also to remain compatible with potential dynamic evolutionary approaches. The Design Games Framework resides on a level between how design is understood and how it is implemented through computational (evolutionary) means, and aims to be a research tool for the exploration of dynamic evolutionary design. Although tackling a practical task is not amongst the aims of this study, the framework will be applied through a toy problem, to exemplify a possible mapping between the rather abstract concepts of the framework and the operators of a specific EA. The study does not aim at solving the problem of dynamic evolution, either. Yet, it will enable us to indicate what kinds of additional technologies would be required, if evolutionary design is our target.

2. Design games model and the framework

With their un(der)defined problem areas, solution procedures, and unexpected products, design processes often defy anticipation (Lawson, 2005; Dorst, 2006; Cross, 2006). The illustration in Figure 1 attempts at capturing these characteristics of design processes, particularly the dynamic interleaving and hierarchies of design actors and actions, while trying not to neglect the assumptions, obscurities, and unknowns. The illustration depicts a design situation with blurry components and indefinite stages, as a complex, collaborative process. The elements in Figure 1 represent fictive events that take place within a design situation through time. Areas that pertain to the relatively unified design processes are delimited by thick dotted borders. Each of these processes is extended over a period, occasionally overlapping with others. Within each distinctive design process, a series of “design game” areas are located. These represent the different subtasks within an overall design process and are drawn by continuous thin lines. Other elements such as tools and agents function together to constitute these games. Each element is a unique construction, indicated by a unique symbol. It appears and disappears at a specific period during the design process in order to carry out an aspect of the process. The relationships between these elements are transient.

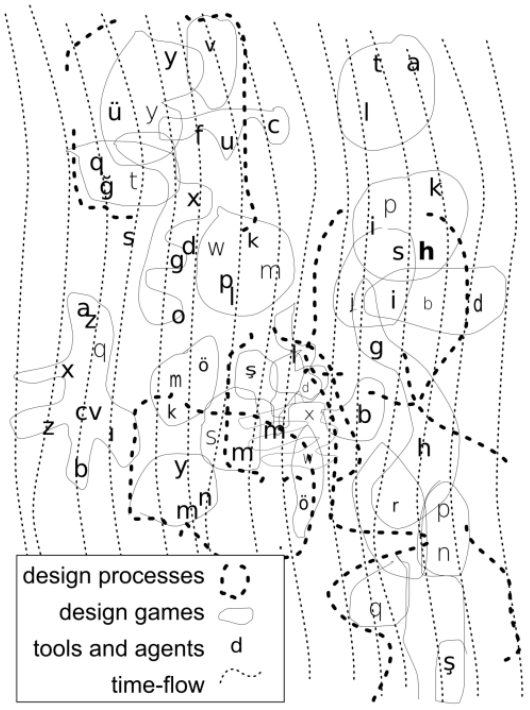

Figure 1. *A depiction of design processes through "design games" (Adapted from Sönmez and Erdem, 2009).*

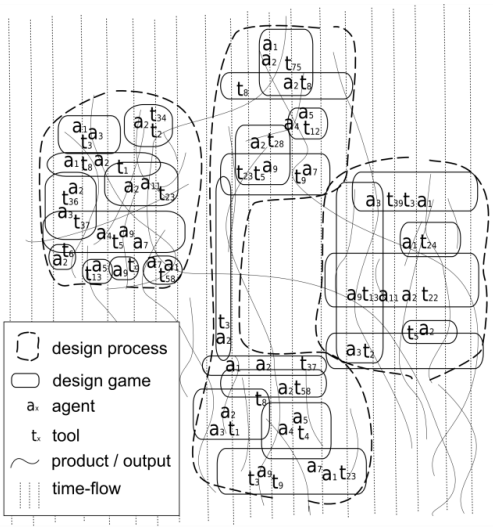

Figure 2. *The Design Games Model (DGM) (From Sönmez and Erdem, 2009).*

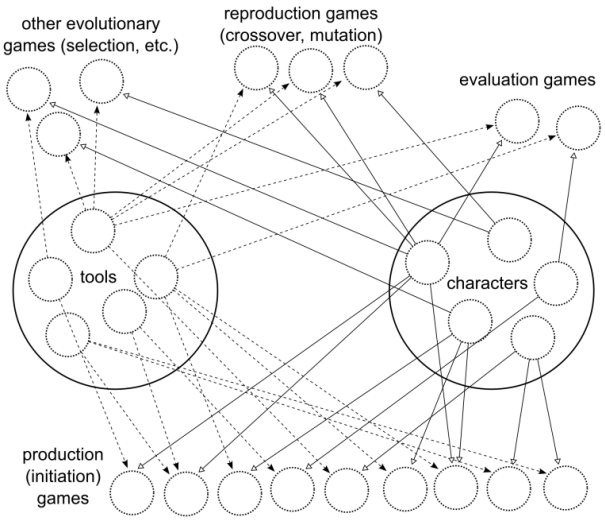

Figure 3. *Basic scheme for Design Games Framework (DGF).*

Through a series of simplification and specification operations on this illustration, the Design Games Model (DGM, Figure 2) has been proposed by Sönmez and Erdem (2009), with rather practical aims. In DGM, design processes and individual design games are clearly distinguished. There are a number of distinct tools and agents that are assumed to be reusable (indicated by letters, t and a). The agents act as carriers for techniques that would guide the generic reusable tools within specific contexts. The objects that correspond to the design proposals and intermediary outputs are represented through trajectories. These outputs may be partial or holistic solution proposals in many forms (drawings, models, etc.) and are assumed to be transformed through design games.

Claiming that the DGM is a suitable conceptualization for carrying out evolutionary design, this paper proposes a general evolutionary design framework, i.e., the Design Games Framework (DGF, Figure 3), with an aim to discuss the possibilities for a dynamic evolutionary design approach.

In the DGM, design processes are depicted as combinations of design games, which in turn are combinations of tools and agents. In the corresponding evolutionary framework, basic constituents are characters (agents), tools, games, and objects (partial or whole solution proposals), which are taken as atomic units. A mapping from the model to the framework is given in Figure 4. Tools and characters are the basic building blocks within the DGF. Tools are generic computational operators that may transform a given state and the characters are specifications that may guide or govern the usage of the tools. A tool (generic operator) will be transformed into a determinate operator (design game) only when it is specified through a character (or agent), within a specific context. This unified functioning of at least one character and one tool constitutes a design game. This separation of the tools from how they will be used aims at making dynamic and contextual definition of evolutionary operations and evaluations possible, at least in principle, as these evolutionary operations will be conceptualized as design games. A game itself may be part of other higher-level games and may include sub-games. For the sake of generality, these definitions reside

on an abstract level and no further definitions will be provided. DGM depicts a collaborative approach and the DGF attempts at defining a platform for the collaboration of an indefinite number of both human and non-human agents.

Evolutionary approaches frequently employ human designers within interactive evaluation processes, which obviously brings the human back into a tiresome procedure and therefore diminishes the value of the partial automation provided by the evolutionary approach. An ability to solicit human designers' preferences and domain-specific procedural knowledge in a less tiresome and effective manner would increase the value of evolutionary design approaches. This is one of the reasons behind the idea of virtual characters, i.e., capturing the stylistic or procedural preferences of a human designer within computational constructs. Another idea is to develop a multitude of such definitions, to be used as a repository of styles.

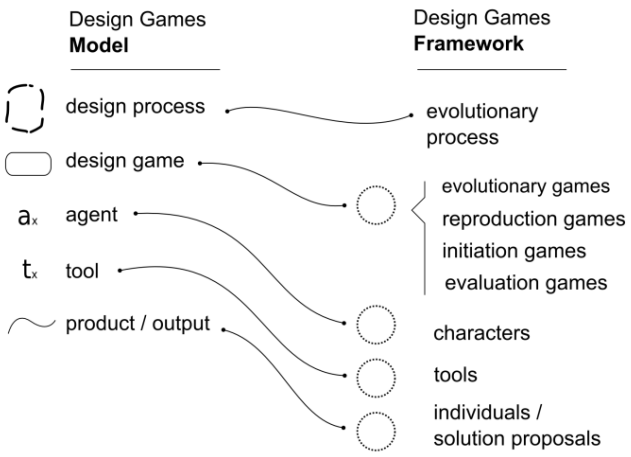

Figure 4. *A mapping for the corresponding elements of Design Games Model and Design Games Framework.*

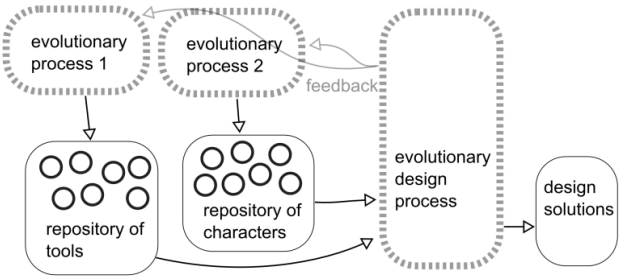

Figure 5. *A multi-level co-evolution / learning model for DGF.*

The aim of the DGF is to devise dynamic EAs that correspond to dynamic problem settings, even when task representations remain static. In practice, selection of the tools with respect to the requirements of a specific context and matching of these with appropriate characters may only be possible with a strategical intelligence, capable of monitoring and evaluating a represented context and the progression of an evolutionary process. We will indicate several evolutionary options for such an aim. The first option is to evolve the tools and characters of a system through a multi-level co-evolutionary process. In the simple multi-level system depicted in Figure 5, tools and characters are being evolved on the first level, while an evolutionary design process, which uses combinations of these tools and characters as operators (i.e., as games), is taking place on the second level. This kind of algorithm can be interpreted as a learning system, which continuously develops itself by learning new tools and characters on the run. In such a system, the task definition for the design process will remain static, the evolutionary processes that produce the tools and

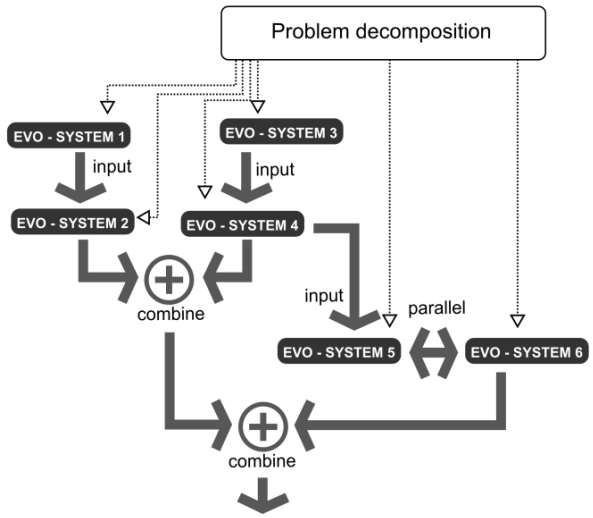

Figure 6. *Decomposition through productions.*

characters will have to become adaptive through constant feedback, and the main evolutionary process will be in constant change, which would render the evolutionary system partially dynamic (Figure 5). Note that what makes the main EA dynamic is the constant redefinition of its operators.

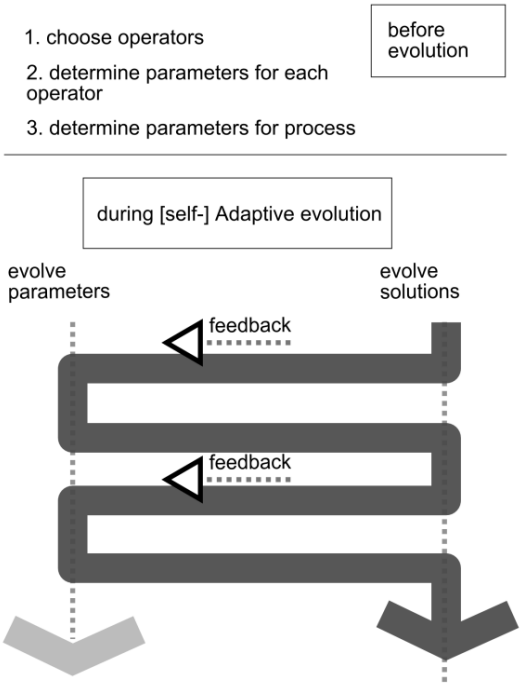

Figure 7. *Self-adaptive evolution.*

The second option for dynamism is to decompose a design process into separate productions and to match each of the productions to a specific EA (Figure 6). In the previous option, the EAs were operating simultaneously as a single dynamic system. In this second option, each of the static EAs operate in isolation, to be related with the others only through their inputs and outputs. In other words, different evolutionary processes may be combined within a single workflow, so that the output of one may become an input for another. This approach is simply exemplified in the application that will be presented at the end of this paper, where the stamps developed during early evolutionary trials are used as tools during the latter stages of a pattern generation task. A parallel strategy would be to develop different constituents of a product, such as parts of a chair or a table separately, to be combined later. There may be an infinite variety of how separate evolutionary processes could be interacted, which generates a potential, not for dynamic EAs, but, on a higher level, for dynamic and complicated contextual reorganizations of a series of ready-made static EAs.

A more integrated hierarchical strategy is self-adaptive evolution (Eiben and Smith, 2003), where parallel evolutions are brought together into one integral evolution (Figure 7). In a simple case, the parameters (characters in DGF) that govern operators are evolved together with the products. There are several levels where adaptivity can be implemented. First, adaptivity may concern the whole process and use feedback from the health of the process, which is, however, not straightforward to determine. Second, to each candidate that is being evolved, its own operator parameters can be assigned. Third, each component of each individual may be assigned its own parameters. Therefore, it can be claimed that, in adaptive evolution dynamic characters are assigned to each process, each candidate, or each component. In the last two cases, the feedback procedure is implicit. The underlying idea is, the better the individual produced, the better the operators should have been that have created it. In these variants, the process operators will stay fixed, however, the reproduction operators (crossover and mutation) can be co-evolved together with the products in an effective manner. An example application in graphic arrangement generation can be examined in Sönmez, Erdem, and Sarıyıldız (2010).

3. A mapping from the elements of DGF to evolutionary operators

To demonstrate the applicability of the DGF, it should be shown that the conceptual elements of the DGF could be mapped to basic evolutionary

operators. The following mapping and evolutionary application will not be a demonstration of the applicability of the dynamism of the DGF, because such a demonstration requires, in addition to the above mentioned evolutionary strategies, further artificial intelligence techniques to carry out contextual and dynamic evaluation that demand separate studies.

There are several core operations in an EA, such as initiation, genotype-phenotype mapping, evaluation, mutation, and selection. For the application, to each of these operations, specific tools and characters will be assigned, so that each operator will be conceptualized as a design game, i.e., operators of an EA will be constituted as design games. Figure 8 illustrates the types of example characters and tools, and how evolutionary operators will be constituted from these as design games.

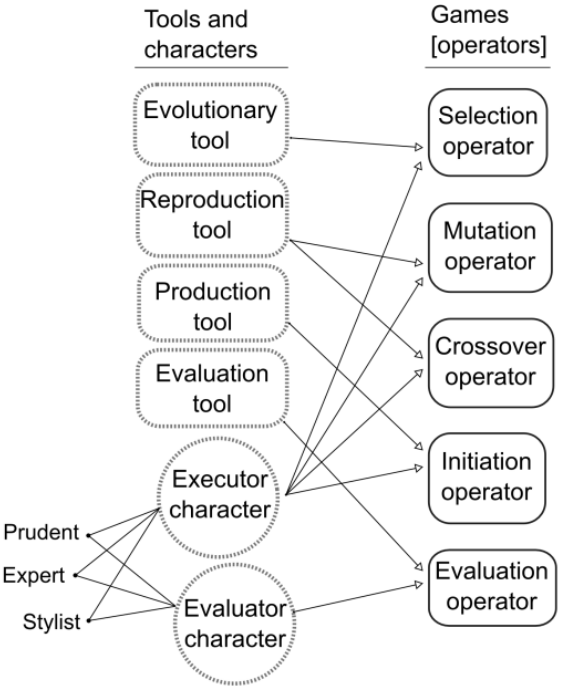

Figure 8. *Constitution of evolutionary operators as games.*

In DGF, each character type may involve sub-types, such as different characters to guide a specific operation. Two basic types of characters are given in Figure 8. The executor characters define how the evolutionary operations will be carried out, while the evaluator characters guide the evaluation tasks. There may be a series of executor and evaluator characters, which will constitute alternatives for each operator. Likewise, each type of evaluation may be carried out by a separate character. Each operator within an EA may be constrained by using 'prudent' executor or evaluator characters, in order for them to comply with already known specifications. Additionally, 'expert' characters can be introduced within an EA, to implement already known procedures. There may be 'stylist' characters, which can either push (execution), or pull (evaluation) a generation of candidates towards a desired direction. In the executor type, a stylist character may apply a procedure in a definite manner, to produce a desired effect, while in the evaluator type it can measure a candidate's compliance to some visual specifications. In principle, an infinite number of types and sub-types of characters can be defined, such as, novel, subtle, bold, delicate, colorful, mad, traditional, etc. These characters can be used together in a collaborative environment, serving as a team of customizable virtual agents. Similarly, there may be a pool of alternative tools for each operator. As claimed above, for design problems, evolutionary operators cannot be specified and fixed before the evolutionary process starts; rather, they have to be determined on the run with respect to the contextual formations. If an intelligent technology capable of contextual evaluation was available, an intelligent choice amongst the alternative characters and tools could serve for the required dynamism.

The DGF will be applied on a toy task, which concerns the generation of a series of desktop icons for the system. A candidate icon is produced by the application of a series of graphic stamps, through plain, transparent, or gradient color definitions, over a canvas (Figure 9). These stamps can be

formed by graphic patterns and texts. The task is to develop icons that conform to the style definitions of the evaluator character. Following this task definition, a limited inventory of characters and tools are developed.

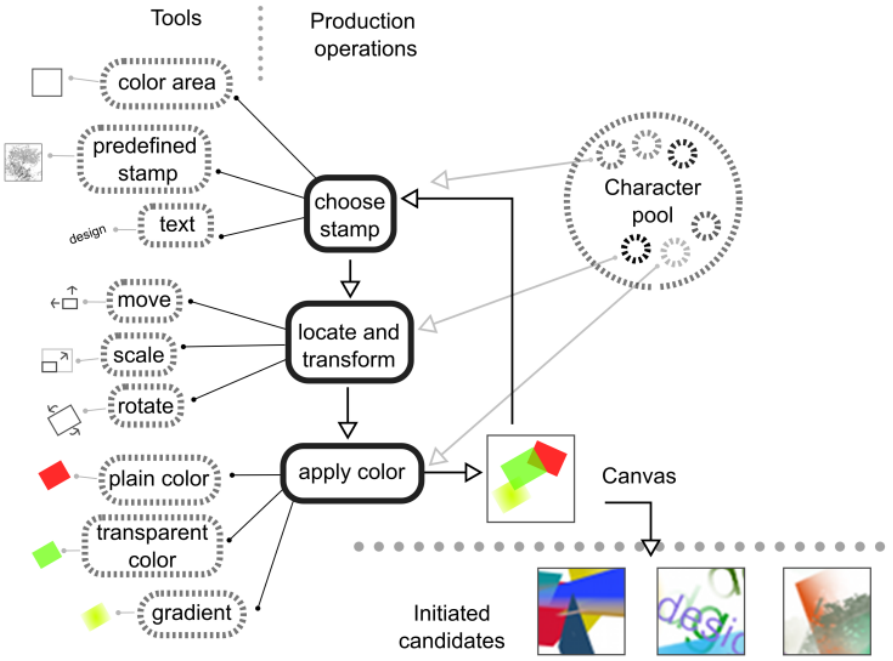

Figure 9. *Representation and initiation for the desktop icon task.*

Characters hold a set of guiding parameters that stochastically determine the behavior of a tool. Most tools allow" for partial randomization, so that the exact implementation values for two applied (determined) games that use the same tool and character will be different. Therefore, instead of the generic character, these values have to be stored within the genotype of an individual. The genotype of a candidate (i.e., individual) is a variable length chromosome, which holds a stack of initiation operations (i.e., production games) (Figure 10). Each gene holds the implementation information of a specific operation, i.e., one or more tools and corresponding implementation values. An individual is produced (genotype-phenotype mapping) by sequentially applying all determined games within the chromosome. With this genotype, it is straightforward to delete or add new genes or to apply crossover.

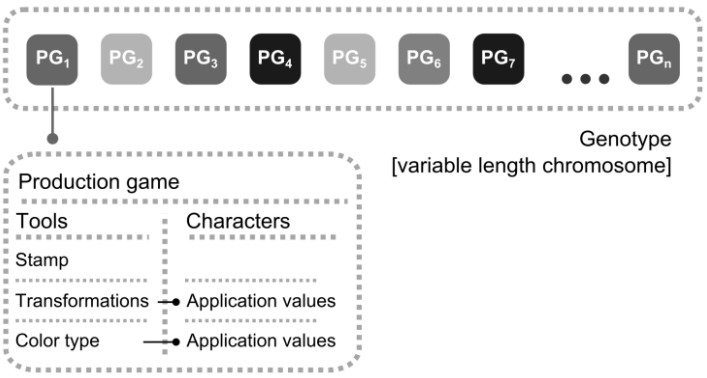

Figure 10. *Genotype.*

The production tools are grouped into three types (Figure 9). On the first level, there are alternative stamp tools (color area, pattern stamp, and text). On the second level, there are matrix transformations (translate, scale, rotate). On the third level, color application tools are situated (plain color, transparent color, and linear gradient).

Executor and evaluator character types are separated. The example evaluator character describes guidelines (in the form of thresholds and coefficients) for rating several visual characteristics (Figure 11). The example executor character specifies which patterns will be used and within which transformation limitations. Additionally, the collection of process parameters for the evolutionary process operators can be interpreted as a fixed, deterministic executor character.

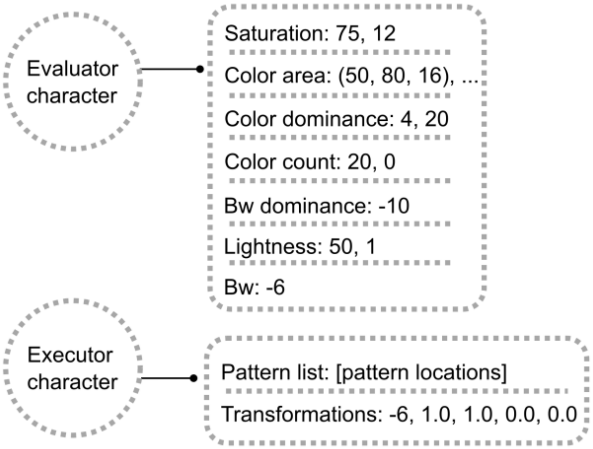

Figure 11. *Character examples.*

The evaluations are carried out through the phenotypic representations. Therefore, several experimental image analysis tools have been implemented to enable the evaluations for high-level visual characteristics. For each phenotype image, using a histogram of color values the following analyses are carried out:

- List most frequent colors.
- Detect whether an image is almost completely black or white.
- Find largest color areas and dominant colors (The areas concern similar colors, determined according to closeness of the hue values).
- Measure if black or white is amongst the most dominant colors.

Once analysis of an image is completed, the results are transferred to the evaluation tools. These tools use this information for additional procedures to rate the fitness of an individual. The different types of evaluations are converted into a single fitness value through weighted aggregation, according to the parameters defined by the evaluator character. The evaluation procedures are as follows:

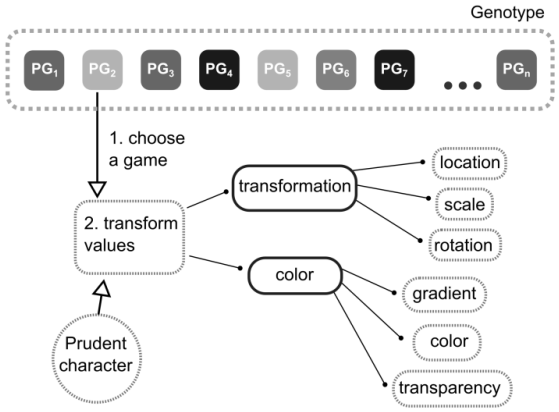

Figure 12. *Parameter mutation.*

- If the image is almost black / white, add specified award / penalty.
- If the character has the option black / white dominance, and if black or white is amongst the dominant colors, add specified award / penalty.
- If color count is below / above the threshold, add award / penalty.
- If the number of dominant color areas in an image is below a threshold, add award / penalty.
- Check each of the most dominant color areas, whether largeness of this area is between specified ratios with respect to the canvas area and add award / penalty for each.
- Using the list of dominant colors, check for the saturation and lightness values of each color. If results are above / below specified thresholds, add award / penalty.

There are three mutation (reproduction) operators:

1. Insert mutation, inserts a randomly generated production game to a random position within the chromosome of an individual.

2. Subtract mutation, deletes a randomly chosen game from the chromosome.
3. Parameter mutation, randomly changes several values of an existing game, such as position, size, transparency, or color (Figure 12).

These mutations are carried out within specified constraints, so that they are kept within meaningful dimensions in relation to the canvas. This is a type of prudence filter, and is controlled by the process character.

For recombination, two crossover operators are implemented. In one-point crossover (Figure 13), a random point is determined in each parent chromosome and the first part of the first chromosome is recombined with the last part of the second, while second part of the first is recombined with the first part of the second.

In two-point crossover (Figure 14), two indexes are determined in each chromosome, and the portion of chromosomes between the indexes is exchanged.

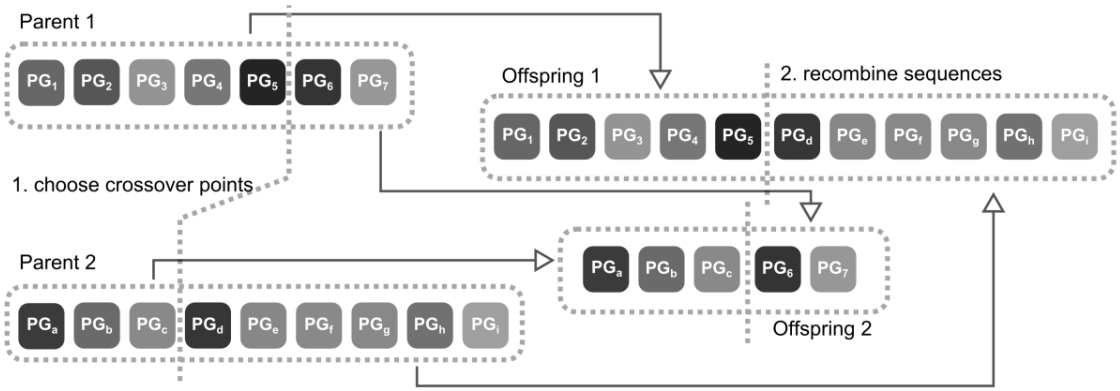

Figure 13. *One-point crossover.*

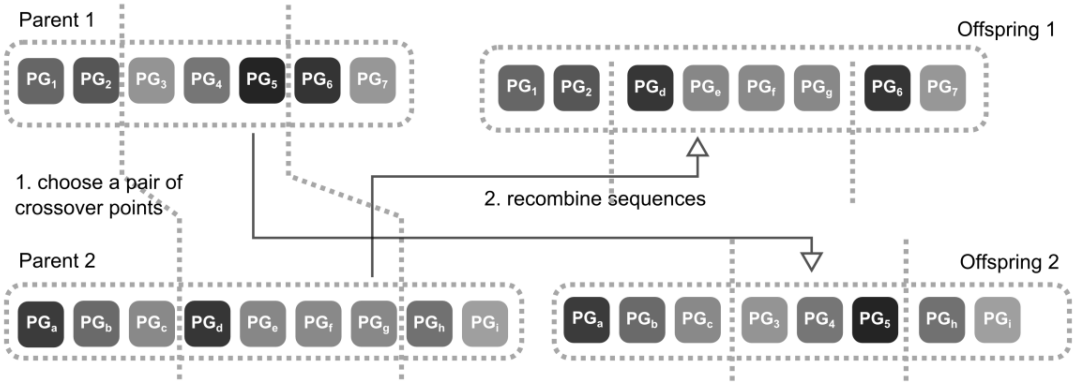

Figure 14. *Two-point crossover.*

The only implemented selection operator is tournament selection. Two parents have to be chosen for the generation of each couple of new candidates. In the implemented tournament selection mechanism, for each parent, two individuals are randomly selected from the population. Then

these are compared in terms of their fitness values and the better one is selected as the parent. Selections for the creation of the new generation are likewise done through tournaments of two.

The evolutionary process starts with the creation of a set of candidates (i.e., individuals) through the selection and application of a series of production games for each candidate. This process is called initiation and the first set of candidates is called the first generation. A series of process parameters define problem-specific aspects and fine-tune the behavior of an EA. In this application, the generation count (which is used as the stopping criterion), the number of individuals at one generation, the quantity of offspring (specified separately for crossover and mutations), width and height values for each candidate image, crossover and mutation methods (tools, characters, stamps, and texts) are fixed before the process begins; therefore, the EA is static. Figure 15 describes the basics of the evolutionary process as implemented.

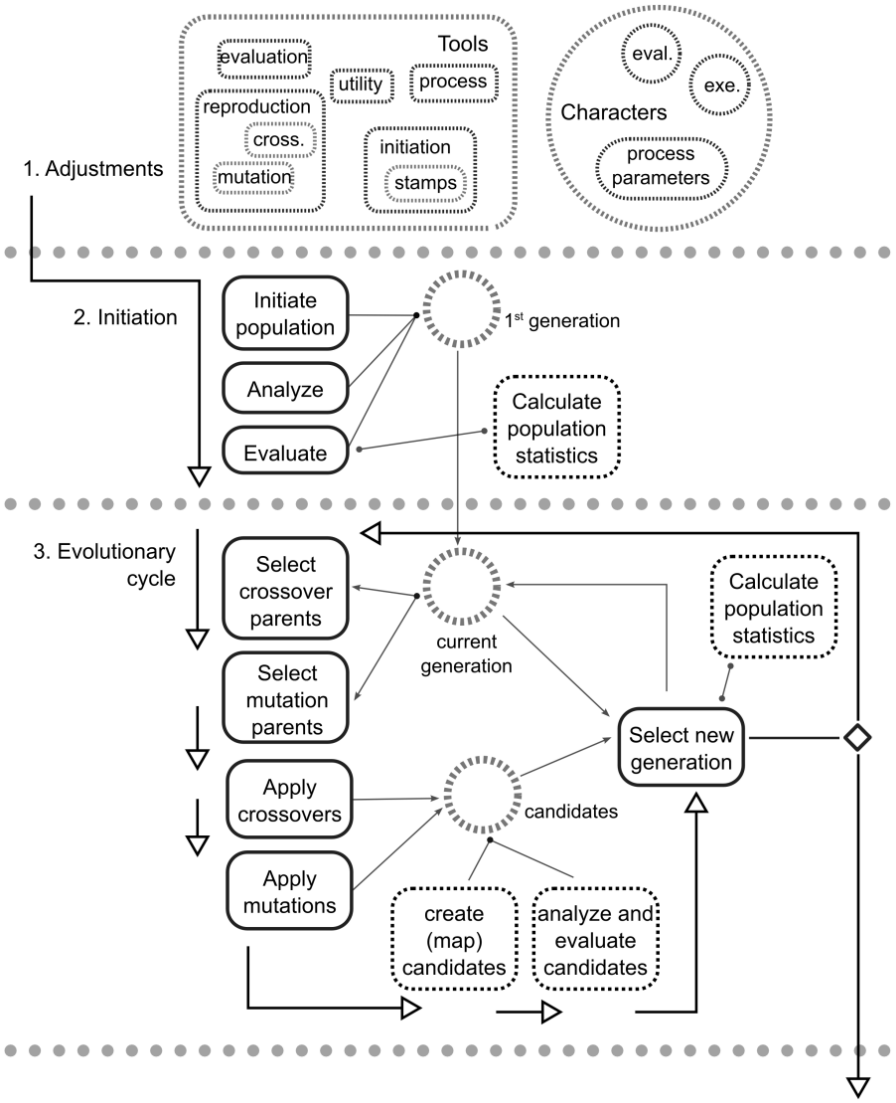

Figure 15. *Evolutionary process for the DGF application.*

As described above, the evaluations are based on several visual qualities. Each of these qualities defines a separate objective, and the multitude of objectives is reconciled with a weighted aggregation method. Each rating for each quality is multiplied by a coefficient, so that the different ratings are scaled according to user-defined preferences. Using combinations of desired values and coefficients for these qualities, different evaluator characters can be defined for the evaluation of candidate images.

4. Presentation of the results

Several sets of trials have been carried out for the desktop icon task. Starting with only plain color areas, other types of stamps (transparent areas, gradients, pattern stamps, and texts) are gradually involved within trials. For the creation of complex stamp patterns, a two-level hierarchical approach is used (Figure 16). On the first level, a set of patterns are manually created. Through a series of evolutionary runs that use these patterns beside other stamp tools, several resulting images are obtained. Some of these images in turn were converted into stamps and added to the pool of pattern stamps for further trials.

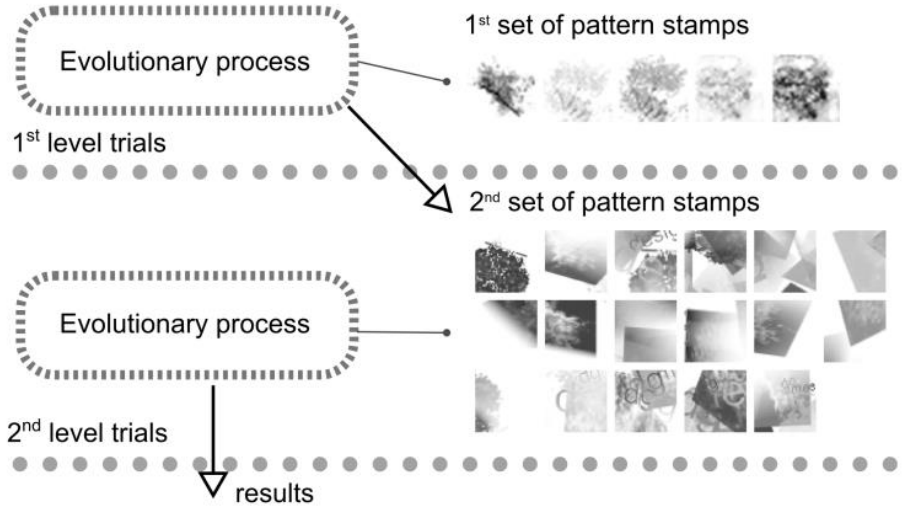

Figure 16. *A simple example of hierarchical evolution.*

Figure 17 presents an example evolutionary run for the icon generation task. The character definition specifies a threshold for number of separate colors. The number of colors is not calculated from absolute RGB color mode values. Instead, the HSL color mode equivalent is found for each color value. The hue value is within a circular range of 0-255 points. For practical reasons, two colors are assumed same, if they are at most 20 points away from each other. In the example, if there are more than 200 separate colors, a penalty is applied. After a list of most frequent colors is found, the saturation test is applied to each, which requires the saturation of a color within specified thresholds. This example demands medium range saturation values. The evaluator character defines three tests for color areas. Each color area is given an award or a penalty, if the ratio of its area to the whole canvas is within a specified interval. The example definition prefers large (0.5 to 0.8 of canvas) and medium sized (0.3 to 0.5 of canvas) color areas, and punishes small ones (0.1 to 0.3). Color dominance is not related to color areas, but with absolute frequency of a color, which might be dispersed throughout the image. The example character rewards high frequencies and

punishes frequencies below a ratio of 0.2. These definitions target kinds of images that have a few dominant color areas and at the same time try to diminish the range of different hues, and the number of small color areas, i.e., artifacts. The last important definition concerns the lightness value, where darker colors are preferred (over 60, within a range of 0-100). The executor character defines the application and process parameters. There is also a prudence constraint, which limits the rotation of the text stamps within 20 degrees. Two of the mutation tools are used in an interchangeable Manner with equal probabilities of selection. This mechanism simply exemplifies the selection of a tool from a pool of tools, to be used together with a character within a design game. From the process graph (Figure 17), it can be observed that the population has converged within 21 generations. The success of the evaluator character can be visually assessed by comparing the randomly generated first generation with the last one. As can be seen, the target of obtaining images with a low range of hues and with a small number of large color areas that are in medium saturation and medium-to-dark tones is largely achieved in this example.

The success or health of an EA can be assessed through the process graphics, which illustrate the progression of average, minimum, and maximum fitness values for each generation. However, this type of assessment does not demonstrate, whether the character approach worked or not. Thus, for our application, it was necessary to be able to visually assess, whether, through the evolutionary process, desired characteristic have been achieved or not. Such visual assessment is possible for only the rather distinctive character types. For these reasons, the evaluator character that is used for the trial (Figure 17) defines variations for rather bold and colorful images, with a few large, medium-to-high saturation color areas. For the same reason, although it was possible within the implementation to use several characters simultaneously as alternatives, it would be difficult to assess which result was associated with each of the characters. This necessitated the use of a single evaluator and a single executor character for each evolutionary run.

Additionally, it proved difficult to define subtle visual characteristics of an image by a parametric approach. The small features and the subtle variations of color gradients proved important for attaining a desired image, which is almost impossible to manually define through parameters. A more practical and possibly more dynamic idea could be an image-based evaluation approach (an example can be found in Sönmez, Erdem, and Sarıyıldız, 2010).

5. Conclusions

This paper presented a general, open-ended, and flexible evolutionary design framework called the Design Games Framework, through which, several options for dynamic evolutionary systems have been explored. A simple application demonstrated how the framework could be used for a working EA. Through this application, firstly, tool, character, and game constructs are implemented and illustrated. Secondly, a simple hierarchical evolutionary procedure is exemplified. Thirdly, although on a primitive level, it is shown that virtual characters can capture stylistic definitions, and that these can be used for evolving images. An evolution in line with the evaluator character is identifiable; however, given the simplicity of the analysis and production tools, attaining impressive results was not a target.

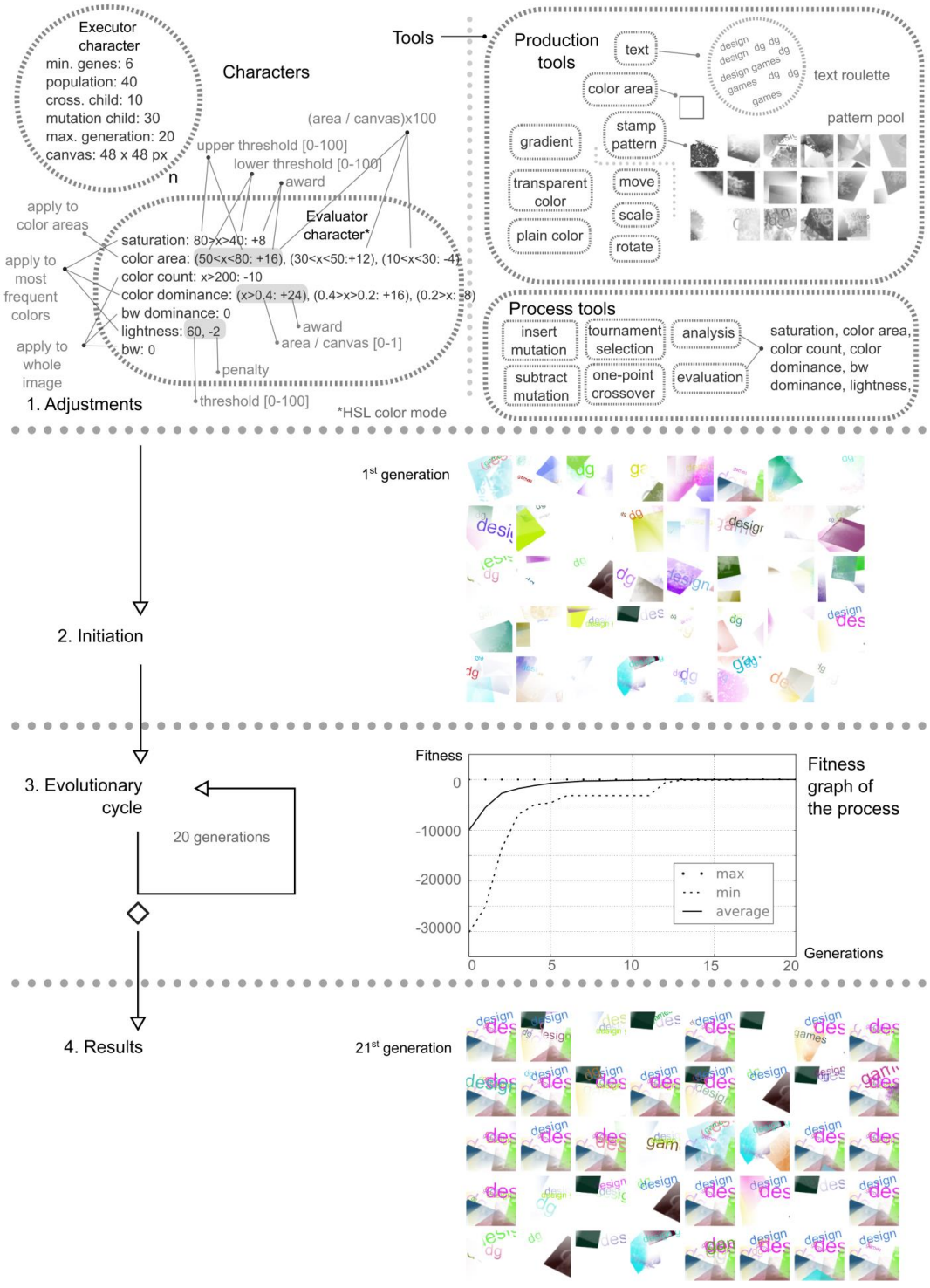

Figure 17. *An example run for the icon generation task.*

The only type of character collaboration was amongst the evaluator and executor characters, which does not demonstrate the cooperative operation of alternative characters for the same operator (or game). The implemented system is the minimum design system that could illustrate the basic aspects of the DGF. The application does not include large tool and character

inventories or elaborate analysis and interface tools, because, utility of these could only be demonstrated after additional intelligent technologies are incorporated. Further studies will illustrate the application of dynamic evolutionary strategies within more complicated problem settings.

Acknowledgements
This study has been supported by TUBITAK (The Scientific and Technical Research Council of Turkey) and Istanbul Technical University "Scientific Research and Development Support Program".

References

Alexiou, K., Zamenopoulos, T., and Johnson, J. H. (2009), Exploring the neurological basis of design cognition using brain imaging: some preliminary results. **Design Studies,** 30 (2009) 623-647.

Ang, M. C., Chau, H. H., Mckay, A., and De Pennington, A. (2006), Combining evolutionary algorithms and shape grammars to generate branded product design. In J.S. Gero (Ed.), **Design Computing and Cognition '06**, 521–539.

Caldas, L. G. (2003), Shape generation using pareto genetic algorithms. **CAADRIA 2003**.

Caldas, L. G. (2005), Three-dimensional shape generation of low-energy architecture solutions using Pareto GA's. **Proceedings of ECAADE'05**, Lisbon, September 21-24, 2005, pp. 647-654.

Caldas, L. G. (2006). GENE_ARCH: An evolution-based generative design system for sustainable architecture. I. F. C. Smith (Ed.), **EG-ICE 2006**, LNAI 4200, pp. 109 – 118, 2006.

Caldas, L. G. (2008), Generation of energy-efficient architecture solutions applying GENE_ARCH: An evolution-based generative design system. **Advanced Engineering Informatics***,* Volume 22, Issue 1 (January 2008).

Caldas, L. G. and Norford, L. K. (2002), A design optimization tool based on a genetic algorithm. **Automation in Construction**, 11 (2002) 173–184.

Caldas, L. G. and Norford, L. K. (2003), Genetic Algorithms for Optimization of Building Envelopes and the Design and Control of HVAC systems. **Journal of Solar Energy Engineering**, August 2003, Vol. 125.

Caldas, L. G. and Rocha, J. (2001), A generative design system applied to Siza's school of architecture at Oporto. In J. S. Gero, S. Chase and M. Rosenman (Eds), **CAADRIA2001**, Key Centre of Design Computing and Cognition, University of Sydney, 2001, pp. 253-264.

Chouchoulas, O. (2003), **Design Shape Evolution: An Algorithmic Method for Conceptual Architectural Design Combining Shape Grammars and Genetic Algorithms**. Phd dissertation, Centre for Advanced Studies in Architecture Department of Architecture and Civil Engineering University of Bath.

Chouchoulas, O. and Day, A. (2007), Design exploration using a shape grammar with a genetic algorithm. **Open House International**, Vol 32, No.2, June 2007.

Cross, N. (2006), **Designerly Ways of Knowing.** Springer,London.

Da Silva Garza, A.G. and Lores, A. Z. (2008), An evolutionary process model for design style imitation. In J.S. Gero and A.K. Goel (Eds.), **Design Computing and Cognition '08**, Springer Science + Business Media B.V. 2008.

De Jong, K. A. (2006), **Evolutionary Computation: A Unified Approach**. The MIT Press.

Dillenburger, B., Braach, M., and Hovestadt, L. (2009), Building design as an individual compromise between qualities and costs a general approach for automated building generation under permanent cost and quality control. In T. Tidafi, and T. Dorta (Eds.), **Joining Languages, Cultures and Visions: CAADFutures 2009.**

Dorst, K. (2006), **Understanding Design.** Gingko Press (2nd edition), Corte Madera, CA.

Dorst, K. and Cross, N. (2001), Creativity in the design process: co-evolution of problem-solution. **Design Studies**, Vol. 22, No. 5, pp. 425-437.

Dorst, K. and Dijkhuis, J. (1995), Comparing paradigms for describing design activity. **Design Studies**, 16 (1995) 261-274.

Eiben, A. E. and Smith, J. E. (2003), **Introduction to Evolutionary Computing**. Springer, New York.

Frazer J. H. (1995), An Evolutionary Architecture. Architectural Association, London.

Geigel, J. and Loui, A. (n.d.), **Automatic Page Layout Using Genetic Algorithms for Electronic Albuming**. Research and Development, Eastman Kodak Company, Rochester, NY 14650-1816.

Gero, J. S. and Kannengiesser, U. (2004), Modelling Expertise of Temporary Design Teams. **Journal of Design Research**, 2004 - Vol. 4, No.2.

Gero, J. S. and Kannengiesser, U. (2007), A function–behavior–structure ontology of processes. **Artificial Intelligence for Engineering Design, Analysis and Manufacturing**, (2007), 21, 379–391.

Gero, J.S., Louis S., and Kundu, S. (1994), Evolutionary learning of novel grammars for design improvement. **AI EDAM**, 8(2):83-94.

Gero, J. S. and Schnier, T. (1995), Evolving representations of design cases and their use in creative design. **Third International Conference on Computational Models of Creative Design**.

Damski, J. C. and Gero, J. S. (1997), An evolutionary approach to generating constraint-based space layout topologies. In R. Junge (Ed.), **CAADFutures 1997**, Kluwer, Dordrecht. pp. 855-864.

Dawkins, R. (1996), The Blind Watchmaker. Penguin Books, pp. 43-74.

Gero, J. S. and Kazakov, V. A. (1998), Evolving design genes in space ayout planning problems. **Artificial Intelligence in Engineering**, 12 (1998) 163-176.

Hanna, S. (2005), Automated representation of style by feature space archetypes: distinguishing spatial styles from generative rules. **International Journal of Architectural Computing**, issue 01, volume 05.

Hanna, S. (2006), Representing style by feature space archetypes, description and emulation of spatial styles in an architectural context. In J.S. Gero (Ed.), **Design Computing and Cognition'06**, 2006, 3–22.

Hanna, S. (2007), Defining implicit objective functions for design problems. **GECCO'07**, July 7–11.

Janssen, P. H. T. (2004), **A design method and computational architecture for generating and evolving building designs**. Phd Dissertation, Hong Kong Polytechnic University.

Janssen, P. H. T. (2006), A generative evolutionary design method. **Digital Creativity**, 17:1, 49 – 63.

Jo, J. H. and Gero, J. S. (1998), Space layout planning using an evolutionary approach. **Artificial Intelligence in Engineering**, 12 (1998) 149-162.

Kalay, Y. E. (Ed.) (1992), **Evaluating and Predicting Design Performance**. Wiley, New York.

Lawson, B. (2004), **What Designers Know**. Elsevier / Architectural Press, Amsterdam.
Lawson, B. (2005), **How Designers Think: The Design Process Demystified**. Architectural Press (fourth edition).
Lawson, B. and Dorst, K. (2009), **Design Expertise**. Architectural Press.
Lewis, M. (2008), Evolutionary visual art and design. In Romero, J., and Machado, P. (Eds.), **The Art of Artificial Evolution: A Handbook on Evolutionary Art and Music,** Springer-Verlag.
Liu, H. and Tang, M. (2006), Evolutionary design in a multi-agent design environment. **Applied Soft Computing**, 6 (2006) 207–220.
Rittel, H. W. J. and Webber, M. M. (1973). Dilemmas in a General Theory of Planning. **Policy Sciences**, 4 (1973), 155-169.
Rosenman, M. A. (1997), The generation of form using an evolutionary approach. [online] Available at: <http://arch.usyd.edu.au> (Accessed: September 2009).
Rosenman, M. A. and Saunders, R. (2003), Self-regulatory hierarchical coevolution. **AI-EDAM**, 2003, 17, 273–285.
Ross, B.J., Ralph, W., and Zong, H. (2006), Evolutionary image synthesis using a model of aesthetics. In G.G. Yen, S.M. Lucas, G. Fogel, G. Kendall, R., Salomon, B.T. Zhang, C.A.C. Coello, and T.P., Runarsson, (eds.), **Proceedings of The 2006 IEEE Congress on Evolutionary Computation**, Vancouver, BC, Canada. IEEE Press, 2006, 1087–1094.
Schön, D. A. (1983), **The Reflective Practitioner**. Basic Books, New York.
Simon, H. A. (1973). The Structure of Ill-Structured Problems. **Artificial Intelligence**, 4 (3): 181–201.
Simon, H. A. (1996), **The Sciences of the Artificial**. MIT Press (3rd edition), Cambridge, Mass.
Sönmez, N. O. and Erdem A. (2009), Design Games as a Framework for Design and Corresponding System of Design Games. **Computation: The New Realm of Architectural Design 27th eCAADe Conference Proceedings**, Istanbul (Turkey), 16-19 September 2009, pp. 119-126.
Sönmez, N. O., Erdem, A., and Sarıyıldız, S. (2010), Automated Evaluation and Generation of Graphic Arrangements through Adaptive Evolution. **Generative Art 2010**, Milano, 15-17 December 2010, http://www.generativeart.com/.
Turrin, M., von Buelow, P., and Stouffs, R. (2011), Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. **Advanced Engineering Informatics** [online] Available at: <doi:10.1016/j.aei.2011.07.009> (Accessed: September 2011).
Turrin, M., von Buelow, P., Kilian, A., and Stouffs, R. (2011), Performative skins for passive climatic comfort: A parametric design process. **Automation in Construction** [online] Available at: <doi:10.1016/j.autcon.2011.08.001> (Accessed: September 2011).

Tasarım oyunları:
Dinamik evrimsel tasarım için kavramsal bir çerçeve

Evrimsel hesaplamalar (EH) tabiri, biyolojik evrimden ilham alan bir hesaplamalı teknikler ailesini ifade eder. EH çoğunlukla 'problem çözme' ve optimizasyon alanlarında, özellikle problemlerin kolayca uygulanabilir algoritmik çözümlerinin bulunmadığı durumlarda kullanılmaktadır. Bunların yanında evrimsel yaklaşımlar bazı tasarım ve sanat alanlarını da kapsayan çok çeşitli alanlarda sınanmıştır. Ancak EH'ın tasarım alanlarına çoğunlukla mühendislik alanlarından transfer edilmiş olması

çoğu uygulamada tasarım görevlerinin iyi-tanımlı problemler gibi ele alınması sonucunu doğurmuştur. Çoğu evrimsel tasarım uygulaması, tasarım görevlerini problem çözme ve optimizasyon yöntemleri üzerinden çözebilmek hedefiyle, ya 'iyi tanımlı' alt problemlere odaklanmakta ya da basitleştirilmiş, statik veya performans odaklı tasarlama prosedürleri tariflemektedir. Bu durum pek çok araştırmacıya sorunlu görünmemiştir, zira tasarımı rasyonel bir problem çözme etkinliği olarak tarifleyen, böylece tasarım görevlerinin 'arama' ve optimizasyon yöntemleriyle çözülebileceğini varsayan problem çözme paradigması, tasarım araştırmaları ve kuramları üzerinde önemli bir etkiye sahip olagelmiştir. Ancak, tasarım etkinliğinin tümüyle bu paradigma içinden tariflenebileceğini iddia etmek mümkün değildir. Tasarımcıların karşı karşıya kaldıkları problemler çoğunlukla 'kötü tanımlı' ve 'açık uçlu' problemler olarak nitelendirilmektedir. Çoğu tasarım durumunda, iyi tanımlı, berrak ve bütüncül bir tarife sahip problemler başlangıçta verili değildir; problemlerin karmaşık ve sorunlu durumlar içinden üretilmesi gerekmektedir. Tasarım sürecinde çözümler ve problemler paralel olarak evrimleşir; öyleki, tasarımcının görevi problemlerin çözülmesi kadar bu problemlerin oluşturulmasıdır da. Öte yandan, optimizasyon yöntemleri belirli bir durumun özsel gereklerinin, çözüm süreci başlamadan önce listelenebileceğini ve ölçülebilir bir formda ifade edilebileceğini varsayar. Fakat tasarımcılar, problemin tüketici bir ifadesine sahip olmamanın ötesinde, sayısallaştırılabilir kriterlere de sürecin başında sahip olmayabilirler. Tasarım ürünleri kesin olarak doğru ya da yanlış olmadıkları gibi, bunların birbirleriyle karşılaştırılmaları da yoruma açıktır. Bu sebeplerle, en azından ana kararların artık oturmuş olduğu detay tasarım aşamalarına kadar, tasarım bir optimizasyon problemi olarak ele alınamaz.

Evrimsel tasarım, tasarım süreci boyunca dönüşen bağlamı yeniden değerlendirebilecek dinamik problem tarifleme ve değerlendirme yaklaşımları gerektirmektedir ve bu gerekler henüz evrimsel tasarım uygulamalarında karşılıklarını bulmamışlardır. Bu sebeplerle, evrimsel 'tasarım' uygulamalarının varlığından söz edilebilmesi güçtür. Bu alanda pratiğe yönelik uygulamalardan daha öncelikli olarak, tasarım kuramları ve uygulama arasında kalan bir seviyede, tasarımın kendine has özelliklerini hesaba katacak yeniden değerlendirmelere ihtiyaç duyulmaktadır. Dolayısıyla, bu çalışmada tasarımdaki EH uygulamalarının eleştirel bir yeniden değerlendirilmesinin gerçekleştirilmesi, dinamik evrimsel tasarımın araştırılmasına yönelik bir kavramsal çerçevenin önerilmesi ve bu çerçevenin bir evrimsel algoritma bağlamında işlerliğinin ortaya konması hedeflenmiştir.

"Tasarım Oyunları Çerçevesi" dört temel öğeden oluşmaktadır: karakter, araç, oyun ve ürün. Tasarım süreçleri, tasarım ürünlerini dönüştüregelen tasarım oyunlarının kombinasyonları ile tariflenir. Her bir tasarım oyunu en az bir karakter ve bir aracın bir arada işleyişiyle oluşmaktadır. Araçlar tasarım önerilerini ve tasarım durumunu dönüştürür. Karakterler ise bu dönüşümün tarzını kontrol eder, zira araçlar çok-kullanımlık, jenerik kurgulardır. Araçların jenerik tarifi ile kullanım tarzları arasındaki ayrım sadece çok amaçlı araçların üretilmesini değil, daha da önemlisi, dinamik evrimsel süreçlerin üretilmesini hedefler. Çerçevenin detaylı tariflenmesinin ardından dinamik evrimsel tasarımı mümkün kılacak mevcut evrimsel mekanizmalar tartışılmıştır. Bunlar, çok-seviyeli paralel evrim (öğrenen sistemler), problemin üretim üzerinden bileşenlere ayrıştırılması ve kendi kendine adapte olan evrim olarak sıralanmıştır.

Bir tasarım problemini pratikte ele almak ya da dinamik evrim sorununu çözmek çalışmanın amaçları arasında yer almamakla birlikte, Tasarım Oyunları Çerçevesi'nin görece soyut bir ölçekte yer alan kavramlarının bir evrimsel algoritmanın bileşenlerine nasıl uygulanabileceğini bir örnek üzerinden haritalamayı hedefleyen bir deneysel EH uygulaması da makalenin son kısmında sunulmaktadır. Bu amaçla geliştirilen üretim ve analiz araçları, karakter tipleri, oyun kurguları ve evrimsel hesaplama yaklaşımını oluşturan temsil, eşeyleme, üreme, değerlendirme, mutasyon ve evrimsel süreç detaylı biçimde aktarılmış, gerçekleştirilen denemelerin ayrıntıları ve sonuçları sunulmuş ve tartışılmıştır. Önerilen çerçeve uyarınca daha karmaşık tasarım problemlerine yönelik dinamik evrimsel stratejilerin ve gerekli teknolojilerin geliştirilmesi sonraki çalışmaların konusudur.