Towards an integrated generative design framework

Vishal Singh, School of Architecture and Building, Deakin University, 1 Geringhap Street, Geelong, Victoria 3220, Australia
Ning Gu, School of Architecture and Built Environment, University of Newcastle, New South Wales 2308, Australia

Design creativity techniques encourage divergent thinking. But how well do the existing generative design techniques support this requirement? How can these general techniques be augmented for supporting design exploration and creativity? This paper investigates these questions through a review of five different generative design techniques used in architectural design that includes cellular automata, genetic algorithms, L-systems, shape grammars, and swarm intelligence. Based on the literature on design cognition and the recent theoretical works on digital design thinking, this paper proposes the need for an integrated generative design framework to enhance design exploration support for human designers. Potential challenges and strategies towards developing such an integrated framework are discussed.
© 2011 Elsevier Ltd. All rights reserved.

Keywords: generative design, architectural design, digital design, design cognition, reflective practise

The main incentives for adopting generative design (GD) systems in architecture are to use computational capabilities to support human designers and (or) automate parts of the design process. The ability to explore larger design space and support design generation is one of the main objectives besides achieving efficiency (multiple design instances in limited time), cost reduction (reduced time and labour), optimisation, accuracy, consistency, etc. This paper focuses on the objectives of design exploration and support for architectural design. The paper reviews five commonly used GD techniques in architecture to include shape grammars (SG) (Stiny, 1980), L-systems (LS) (A Lindenmayer, 1968), cellular automata (CA) (von Neumann, 1951; Wolfram, 2002), genetic algorithms (GA) (Holland, 1975) and swarm intelligence (SI) (Deneubourg, 1977; Payman, 2004). A review of the literature suggests that most existing GD systems are developed based on one of these GD techniques. This limitation often compromises the performance of the system in terms of the breadth of design exploration. Although there are overlaps and similarities, each of these techniques appears more suitable than others for specific design tasks. This paper argues for the need of an

Corresponding author:
Vishal Singh
vishal.singh@deakin.edu.au

integrated GD framework that can support multiple techniques. The proposed integrated GD framework is envisioned as an interactive design generation system. Compared to most existing GD systems that are limited to one single GD technique, which often biases the GD process in a constrained direction, the proposed integrated GD framework can provide design triggers at different phases of the design development and assist in design exploration across different pathways.

Sections 1 and 2 present a review of the five GD techniques in terms of their (1) technical factors (2) design factors, and (3) requirements for system development. Section 3 briefly discusses the literature on design cognition and digital design thinking, forming the basis for Section 4, which presents the conceptual integrated GD framework and the challenges and strategies towards developing such a framework. Section 5 concludes the paper with a discussion on the current and potential future work.

1 Literature review of generative design techniques

Though each of the five computational approaches has broader significance across a range of disciplines spanning fundamental sciences to art and technology, the review is confined to their application to architecture design.

1.1 Cellular automata (CA)

CA is a collection of cells on a grid of a specified shape that evolve over time according to a set of rules driven by the state of the neighbouring cells (von Neumann, 1951; Wolfram, 2002). The grid can range from one-dimensional lines to Cartesian grids in arbitrary dimensions. CA has been used for different aspects of architecture and urban design (Herr & Kvan, 2005; Krawczyk, 2002b). Based on the literature (Krawczyk, 2002b), Table 1 lists some of the basic terminologies used in 3D CA.

CA is naturally context-sensitive as the generative process is directed by the states of the neighbouring cells. Design constraints are implemented from

Table 1 Basic terminologies in (3D) CA

Terminology	*Description*
Cell	Cell is the smallest operationalised unit to define a space (3D/2D). The unlimited lattice of cells can describe the universe (3D space/2D plane).
Cell state	Cell state describes the current status of the unit space. For example, a space can be characterised as empty or full. Cell state can de defined as per the design requirements. Cells can reach stable states such that they are not perturbed by further changes to their neighbourhood. Cells can also be unstable and subject to changes as an effect to the changes around them.
Neighbourhood	Cells sharing some pre-defined spatial proximity to a given cell are considered as its neighbourhood. For example, cells sharing at least one face/edge/vertex of a cell can be termed as neighbouring cells. Choice of neighbourhood definition/condition is subject to problem requirements and desired design characteristics.

bottom-up governing the local behaviours of each cell. As a result, the outcomes of CA are often complex and difficult to predict. The basic components for composing design using CA are cells. Although there are mechanisms to transform cells during the application, in general, cells and their available operations are limited and are constrained by the type of grid adopted.

1.2 Genetic algorithms (GA)

GAs and genetic programming are evolutionary techniques inspired by natural evolutionary processes. GAs, as Gero and Kazakov (2001) state, '… use the analogues of evolutionary operators on a population of states in a search space to find those states that optimise a fitness function. The search space consists of character strings of fixed or variable length (chromosomes or genotypes) composed of the elements of a given alphabet (alleles). The genotype space is mapped onto another (phenotype) search space. The fitness function is defined as a function of a state in the phenotype space.'

In design GAs have been used for diverse purposes such as optimisation (Bullock, Denham, Parmee, & Wade, 1995; Caldas & Norford, 2001; Hybs & Gero, 1992; Mitchell, 1997; Salge, Lipski, Mahlmann, & Mathiak, 2008), space layout planning (Coates & Hazarika, 1999; Gero & Kazakov, 1998; Jo & Gero, 1998), emergence of representation styles (Ding & Gero, 2001) and architectural forms (Caldas, 2001). For example, Caldas and Norford (2001) use GAs with SG to generate façade designs that optimise annual energy consumption. Table 2 lists the commonly used GA terms in design (Ding & Gero, 2001).

1.3 Shape grammars (SG)

An SG is a set of shape rules that can be applied to generate a set or language of designs (Knight, 1999). The application of the shape rules generates designs, and the rules themselves are the descriptions of the generated designs. SGs can be used as design tools to generate design languages or as analysis tools to understand existing designs. Stiny (1980) defines four basic components of an SG:

1. A finite set of shapes
2. A finite set of symbols

Table 2 Commonly used terms in GA with respect to design

Terminology	*Definition*
Gene	The smallest unit of a genotype. *Alelles* are alternative forms of genes.
Genetic codes	Numerals or alphabet letters used for coding in a genotype.
Genetic description	A description employing genetic terms. The genetic operations used in evolutionary systems can then be carried out on this genetic description.
Genetic structure	A set of genes with a certain order or relationship.
Genotype	The genetic constitution of a design, rather than its physical appearance.
Phenotype	The observable properties of a design, its form.

3. A finite set of shape rules
4. Initial shape

A design generated by an SG is viewed as 'elements in relations'(Stiny, 1990). In SG, design generation is basically to change (add, subtract or replace) the 'elements' and define or alter the 'relations' among the 'elements' via shape rules. An SG enables different designs that share a similar style to emerge by alternating the sequence of shape rule application. Examples of SG implementations include the Palladian grammar (Stiny & Mitchell, 1978), Mughul Gardens grammar (Stiny & Mitchell, 1980), Prairie Houses grammar (Koning & Eizenberg, 1981) and Siza Houses grammar (Duarte, 2005).

The critical challenge in developing a grammar is to produce designs that meet the design goals or constraints. There are two main approaches (Knight, 1998, 1999).

1. Incorporate foreknowledge into shape rules so that the generated designs could meet the given goals. This approach constrains the grammars and controls the shape rule application to increase the predictability of the outcomes. This approach requires adequate design knowledge while developing the SG.
2. Allow the SG to generate designs without being constrained. An automated search and test device is then applied to search through the design space, test and select the desired solutions. This approach can start with inadequate design knowledge while developing the SG.

1.4 L-systems (LS)

LSs (Lindenmayer, 1968) are mathematical algorithms known for generating factual-like forms with self-similarity that exhibit the characteristics of biological growth. LSs have been used for a wide range of design problems from simple computer graphics patterns (Palubicki et al., 2009) to complex city planning and simulation (Kelly & McCabe, 2006).

LS are primarily a set of production rules applied recursively through string rewriting. For the purpose of design generation, LS is in principle a design grammar. However, LS have been distinguished from design grammars as being operated on strings - the symbolic representation of the design - rather than directly on the design itself (Parish & Muller, 2001). In a typical LS, design components are firstly symbolised as strings. By applying various string rewriting mechanisms, the representation of the design is generated. In order to visualise or evaluate the generated design, these populated strings are 'remapped' into appropriate forms through graphical interpretations. Whereas, in a typical design grammar, design components such as shapes are directly used in the design rules, and the rule application implies visual and spatial operations on these components. The computer implementation of the design grammar still requires these design components to be represented as symbols in the computational system eventually. In summary, LS are variations of design grammars.

1.5 Swarm intelligence (SI) and multi-agent societies

Agent based models (ABM) are often used to implement social or collective behaviours (Carley, 1994; Macy & Willer, 2002). Agents are software systems capable of acting autonomously according to their own beliefs. Agents in a society can act independently, or interact and communicate with each other to compete or collaborate and collectively achieve their goals. A variety of ABM has been used in design such as works by Kunz, Levitt, and Jin (1998) and Maher, Smith, and Gero (2003).

SI is the property of a system whereby the collective behaviours of unsophisticated agents interacting locally with their environment cause coherent functional global patterns to emerge (Payman, 2004). SI provides a basis with which it is possible to explore collective or distributed problem solving without centralised control or the provision of a global model. Bonabeau, Dorigo, and Theraulaz (1999) define a swarm in terms of self-organisation and stigmergy.

- In self-organisation structures appear at the global level of a system from interactions among its lower level components based only on local information, without reference to the global pattern. The global pattern is said to be an emergent property of the system. As an example, Theraulaz and Bonabeau (1995) demonstrate how 3D lattice swarm can be used to generate building forms. Self-organisation relies on feedback, randomness and interactions.
- Stigmergy refers to the indirect interaction of individuals when one of them modifies the environment and the other responds to the new environment at a later time. Stigmergy facilitates incremental construction and it is often associated with flexibility. When the environment changes because of an external perturbation, the individuals respond appropriately to that perturbation.

Some of the concepts in SI that are relevant to design tasks include:

- Nest building and self assembly: the global process of a swarm of individuals building a structure of residence and (or) using individual building blocks. Deneubourg (1977) and Theraulaz and Bonabeau (1995) have shown the effectiveness of self-organisation and stigmergy in the creation of 3D spaces. GAs have also been combined with SI to generate functional spaces (Theraulaz & Bonabeau, 1995).
- Division of labour and task allocation (Oster & Wilson, 1978).
- Trail-laying and trail-following behaviours whereby an individual is influenced towards a particular location by another individual or a trail (Bonabeau et al.,

1999). These behaviours have potential applications to design agents immersed in virtual worlds for tasks such as layout planning and way-finding.

2 Comparison of five generative design techniques

The following sections review and compare the different GD techniques in terms of the technical, design and system development factors.

2.1 Technical factors

In general, each technique defines a finite set of elements and/or symbols that can be operated upon using a finite set of production rules and/or operators (Alfonseca & Ortega, 2000; Coates, 2004; Devert, 2009; Frazer, Frazer, Liu, Tang, & Janssen, 2002; Lindenmayer & Rozenberg, 1972). Each technique can be used to generate deterministic as well as non-deterministic systems. CA, LS and SG are generally represented in graphical forms that result from transformations and operations (i.e., addition, rotation, subtraction, etc) on the initial elements, whereas the patterns from SI generally result from the autonomy of the agents (e.g., where agents act on the environment such as nest building) or by plotting the agent activities (e.g., tracing agent's way-finding trail) (Coates, 2004; Coates & Carranza, 2000; Thalmann et al., 2004; Yoon & Maher, 2005). For example, Coates (2004) reports a study using SI that investigates 'how can we determine the best position and orientation for paths between buildings on a site, or more generally, given a set of origin and destination points in space, how will people best move between them?'

There have been instances in which CA and LS (Alfonseca & Ortega, 2000) and SG and LS have been demonstrated to be equivalent. Furthermore, there are examples where some of these techniques are used in conjunction with each other (Jackson, 2002; Jacob, 1996). Using SG and CA together, Speller, Whitney, and Crawley (2007) claim that 'the combination of SG for managing the input and CA for managing the output brings together the human intuitive approach (visualisation of the abstract) with a computational system that can generate large design solution spaces in a tractable manner.' Table 3 discusses the five approaches in terms of

1. Components and requirements to define and implement the system, i.e., the basic elements, operatives, and the operational mechanism needed to implement the system.
2. Rule application, i.e., how the rules are applied?
3. Main advantages of the system in terms of GD
4. Main limitations of the system
5. Level of accuracy, which relates to the level of automation and amount of effort required by the user to evaluate the design generated. Table 3 only lists the general practise. However, even for SG and LS it is possible to have systems that are constrained enough to generate valid solutions that

Table 3 Technical aspects of the generative design techniques

	SG	*LS*	*CA*	*SI*	*GA*
Components & requirements	Set of terminal shapes Set of operators Set of production rules Initial shape	Set of terminal symbols Set of operators Set of production rules Initial symbols	Grid/cells Set of state rules Initial cell states	Autonomous agents Knowledge base Set of actions and behaviours, e.g., communication, observation. Objects/Environment the agents interact with and (or) change.	Alleles (building blocks) Chromosomes/Genotypes (combination of building blocks) Phenotypes (solution) Population (set of solutions) Genetic operations Fitness function
Rule application	Usually one rule fires at a time depending on the conditions being matched the LHS.	Usually as many rules are applied at the same time as defined.	Relevant rules fire to change states. Parallel computational process. Hence, facilitates design visualisation at global level.	Usually one rule fires at a time depending on the conditions being matched the LHS.	Usually one operation is applied at a time on a sub-set of the population.
Main advantages	Geometric (visually defined)	Symbolic	Context-sensitive and neighbourhood effects In-built constraints Purely bottom-up Simultaneous, local vs global effects	Modelling cognitive and social behaviour Simultaneous, local vs global effects	Regular design evaluation and improvement Multiple solutions Optimisation Disruptive innovation
Main limitations	Trade–off – constraints vs exploration. Local to global (incrementally local behaviours are lost)	Local to global (incrementally local behaviours are lost)	Constrained by cell geometry/definition Dimensional constraints For finite cells, conditions are different for edge cells.	Requires other design elements to work with but does not usually include design components by its own.	Progress slows down after achieving near optima solutions
Level of accuracy	Trade-off with exploration Often requires post-generation selection by user.	Trade-off with exploration Often requires post-generation selection by user	Contingent on problem representation and solution interpretation Often results in design emergence that are purely bottom-up	Varies with agent detail and modelling considerations	Contingent on problem formulation, i.e., choice of genotypes and fitness function

may not require later validations by the user. However, greater the number of constraints, fewer is the opportunity for exploration and emergence.

In summary,

- CA is particularly useful for its parallel computational processes and simulating context-sensitive growth patterns, e.g., simulating scenarios where multiple factors are mutually affected by each other such as changes in one zone that lead to changes in the neighbouring zones.
- SG and LS are similar and useful for generating patterns incrementally, particularly form-based designs.
- SG and LS may or may not be constrained, unlike CA, which is always constrained and context-sensitive.
- SI is useful for simulating self-organisation among various units. Same as CA, SI also demonstrates parallel computational processes. In SI the parallel processes occur because of the autonomy of agents/units while in CA it is defined in terms of the cell states and neighbourhood conditions.
- GA is based on modifications, combinations, and other operations on the building blocks as well as the solution space. Design exploration and selection occur at each cycle, which is not the case with SG or LS. Genetic operations can generate random designs, and hence, it is easier to model compared to SG or LS where identifying the rules may take time. The challenge in GA is to select appropriate alleles, chromosomes and the fitness function. GA also enables parallel computation but unlike CA and SI where parallel computation occurs at the local level, in GA parallel computation occurs through exploration of multiple solutions at the global level.

2.2 *Design factors*

Some GD techniques are more suited to specific design purposes than others. For example, GAs have mainly been used for optimisation (Mitchell, 1997; Salge et al., 2008) because the quality of solutions tend to increase with each generation.

SG and LS are particularly suited to develop emergent shapes and patterns that arise from repetitions and operations on terminal shapes. Development of SG and LS is an iterative process because it is difficult to visualise the emergent shapes, often requiring iterative trials to identify the desired terminal shapes and production rules. SG and LS typically follow ‘function-follows-form’ philosophy, i.e., once the form is generated, the emergent shape is evaluated for its functionality.

SGs are typically used for generating 2D shapes and compositions, space layouts, and in some cases 3D compositions (Duarte, 2005; Halatsch, Kunze, & Schmitt, 2008; Koning & Eizenberg, 1981; Stiny & Mitchell, 1978). LS are typically used to generate repetitive patterns, fractals and natural organic forms such as plants and fractals (Alfonseca & Ortega, 1996; Lam & King, 2005;

Lindenmayer & Rozenberg, 1972; Palubicki et al., 2009; Smith, 1984), etc. In design, LS have also been used for generating city roads and networks (Parish & Muller, 2001), terrains and textures (Dai & Ozawa, 1997), as well as building forms (Mueller, Wonka, Haegler, Ulmer, & Gool, 2006).

Being context-sensitive CA tends to follow 'form-follows-function' philosophy. The emergent form is the result of the desired functionality. CA facilitates exploration of social effects through simulations being determined by neighbourhood conditions for solving urban design problems, zoning, block design and building massing tasks (Krawczyk, 2002a, 2002b).

SI and ABM can be used in various ways. Agents can be used to test the usability of the design environment such as way-finding and social behavioural patterns. Intelligent, knowledge-rich agents can be used as avatars and design agents that interact with each other, and with their environment as well as with the human designer.

The GD techniques are reviewed in Table 4 according to the following design factors:

1. Suitability to different design phases and design approaches: Each GD approach has its own strengths and limitations, making it more suitable to some aspects of design than the others.
2. Type of design problem: Based on the design generation mechanisms each of the GD approaches have predominantly been used for certain types of design problems.
3. Complexity of design problem.
4. Characteristics of design outcome.

In summary,

- GA is particularly suitable for design optimisation.
- SG and LS are particularly suitable for form and style generation.
- CA is most suitable for supporting bottom-up context-sensitive design processes.
- SI is useful for (1) evaluating design usability, (2) simulating circulation and navigation patterns to inform design development, and (3) generating designs that emerge from self-organisation of autonomous units.
- CA and SI are particularly suited for behaviour-driven design processes, while SG and LS are suited for form-driven design processes. GA allows an undirected exploration of the design space but provides a search through the fitness function to ensure quality assurance.

Though each technique is more suited than others for specific purposes, there have been examples where different technique have been used for similar purposes.

Table 4 Design aspects of the generative design techniques

	SG	*LS*	*CA*	*SI*	*GA*
Design purpose	Design exploration: space layout and visual compositions.	Design exploration: patterns and visual compositions.	Grid-based design: planning/zoning. Usability—based especially context-sensitive design development.	Design usability such as way-finding. Design evaluation and analysis.	Optimisation. Design enhancement/improvement. Simultaneously pursuing multiple design alternatives meeting fitness criteria.
Design approaches	Emergent shapes and patterns. Geometric shapes. Function follows form. Iterative and re-engineering based approaches.	Emergent repetitive patterns, fractals and natural organic forms. Function follows form. Iterative and re-engineering based approaches.	Study of neighbourhood effects, growth patterns, social phenomenon, etc. Form follows function.	User-centric and use-case designs. Study of social phenomenon, e.g., norm creation, way-finding, etc. Function analysis.	Combinatorial and morphological designs. Disruptive innovation.
Design Problems	Architectural styles, objects and patterns. Regulated and rule —based design. Generally 2D, but also examples of 3D.	Roads and networks, terrains and textures. Forms that evolve naturally and tend to be organic. Generally 2D.	Block design and massing, planning/zoning and urban design. Generally 2D, but also examples of extrusion to 3D.	Walkways, public spaces, lobby areas, traffic flow, and so on. Rarely used for geometric design.	Component—based designs, optimisation problems.
Design outcome characteristics	Emergent and exploratory, geometrical. Solutions usually require validation.	Emergent and exploratory, organic and repetitive. Solutions usually require validation.	Emergent and normative, context-sensitive. Usually satisfactory solutions.	Emergent and normative. Usually usability driven.	Optimized, usually satisfactory solutions. Multiple design alternatives in most cases.

2.3 System development factors

In digital design environments, the designer's competence with the tool as well as the designer's capability as a tool builder are critical to effective design generation (Aish, 2003; Oxman, 2006). Hence, system development factors are critical to analyse across the different GD techniques.

CA and SI tend to be easier to model because they are generally defined in terms of states. The rules for potential cell states in CA are easier to identify because the relationships between the neighbourhood cells and potential configurations can be visualised (this does not mean the emergent configurations are easy to be visualised). However, as the neighbourhood cell levels to be considered for potential cell states increase, the number of rules increases exponentially making it harder to define all the cell states (Speller et al., 2007). For example in a rectangular matrix there are only 8 immediate neighbourhood cells giving a maximum possible 2^8 cell states, which is by itself large. If the next level of neighbourhood cells is also considered then the number of cells become 24 giving 2^{24}possible cell states. While identifying and enumerating all possible cell states may become extremely difficult, often the cell states need not to be exhaustively defined. In general, a very few neighbourhood conditions are enough to generate the desired CA. Developing CA and SI is easier because often these approaches are used to explore some social behaviours. Hence, a purely bottom-up approach works such that the resulting emergent behaviour is generally considered useful.

LS and SG are not defined as much in terms of the states as the expectations to generate emergent patterns. Often re-engineering or backward tracking techniques are adopted to define terminal and non-terminal elements, and a set of production rules that may generate the desired pattern. Not all of the results are considered useful, and, hence, developing such systems may take longer and greater number of iterations. The development of LS and SG is rarely a purely bottom-up approach (Tapia, 1999) (Table 5).

In summary,

- SG has mostly been explored manually. Developing SG rules is a time consuming and iterative process. LS are similarly iterative.
- There are more examples of CA implementation because the cell states are easier to identify and implement unless higher order neighbourhoods are considered.
- SI can be implemented with various levels of agent autonomy and intelligence. Often simple reactive agents are used.
- Choice of the alleles and fitness function is critical to developing a useful GA tool.

Table 5 System development aspects of the generative design techniques

	SG	*LS*	*CA*	*SI*	*GA*
User intervention	High user intervention to analyse outcomes. Iterative process.	High user intervention to analyse outcomes. Iterative process.	Purely bottom–up. Low user intervention once cell dimension, state rules and initial states are defined.	Varies with applications. User may need interaction with design agents. Usability test agents may not require user interventions.	Low user intervention once fitness functions, genotypes and termination conditions are defined.
Development challenge	Difficult to foresee emergent shapes. Often requires re-engineering and backtracking. Scaling is a challenge. Emergent shapes may require new set of production rules.	Difficult to foresee emergent shapes. Often requires re-engineering and backtracking. Repetitive. Hence, scaling is usually not an issue.	Relatively easy to design bottom-up, and to identify possible cell states and neighbourhood conditions if only local context is considered. The challenge increases manifold if cell states need to be defined globally.	Varies with applications and level of agent's activities. Agents can range from simple reactive agents to highly sophisticated knowledge-rich agents.	Problem formulation/ representation and choosing the appropriate alleles, genotypes, phenotypes and fitness functions can be challenging.

3 Design cognition and generative design systems

There is a general consensus that design is a co-evolutionary process (Dorst & Cross, 2001; Lawson, 2005; Poon & Maher, 1997; Schon, 1992). Often the designer starts with an ill-defined problem and as the design activity continues the problem and solutions co-evolve and mutually guide each other. During this process, which Schon (1992) describes as a 'reflective practise', the designer's view of the problem and solution is dynamic and changes, as the designer interacts with the task. Gero (1998) adopts a situated stance to explain how the interaction of designers' past experience and the immediate task environment influence the emergent design solutions. A situated view suggests that the same designers may generate different solutions in different situations. Thus, if the designer's situated state can be altered it is possible to create greater variety in the solutions generated. Based on this implicit assumption various creative design techniques such as brainstorming (Osborn, 1957) and TRIZ (Altshuller, 1984) have been developed. These techniques provide triggers and serve as inertia breakers (Savransky, 2000) to modify the designers' state of situatedness, and as an effect facilitate greater design exploration through different viewpoints.

Digital design approaches can provide such triggers and opportunities to change the designer's situated state. As noted by Oxman (2006), digital design thinking and interaction of the designer with digital design medium is a new form of design, different to the traditional design thinking and interaction of the designer with the paper-based design. Using a generic schema of components, relationships and properties of digital design, Oxman (2006) provides a theoretical framework to categorise various classes of digital design approaches depending on the relationships of the designer, the conceptual content, the design process, and the design object itself. Re-emphasising the centrality of the designer, and interaction and reflection as the key mechanisms in design thinking, Oxman (2006) demonstrates that digital design practises provide greater interaction opportunities than the traditional design interaction with the paper based design. Following are some of key points identified by Oxman (2006) that differentiate digital design thinking from traditional design thinking:

- In digital design environments not only does the designer interact with the free form of design (as is the case with paper-based design) but the designer can also interact with
  - The digital constructs (design representations and elements)
  - Interact with the design representations created by a generative design mechanism, i.e, interact with a design representation generated by a set of pre-defined rules or relationships,
  - Interact with the computational mechanisms that generate the design, i.e., specify the rules and relationships.
- Digital design has lead to greater explication of what was implicit knowledge in the traditional design thinking such as

  - Explication of design processes and mechanisms
  - Explication of the design knowledge and the formulations and associations within the design representation
  - Explication of the generative and evaluative knowledge
- The scope of digital design thinking and designer's individual control is contingent on the designer's computational literacy. Hence apart from design expertise, expertise with the computational tools becomes a critical factor.
- Digital design has created new role for designers based on the nature of interaction with the digital design media. The designer's role is not only limited to generating the design but there is a greater opportunity for the designer as a tool builder (Aish, 2003), controlling and developing tools that generate design through scripting and editing the computational mechanisms.
- Digital design provides the opportunity for dynamic representation and abstractions unlike the static abstractions of traditional design, enabling new bases for design thinking.
- The ability to interact with, control and transform the computational mechanisms that generate design has led to the emergence of new forms of nondeterministic design processes that did not exist explicitly in the traditional design thinking.

While Oxman (2006) explains various classes of digital design approaches, the critical characteristic identified with GD tools is the need for an interactive module that provides designers the ability to control and modify the GD mechanisms in order to achieve the desired solutions. Further, Oxman (2006) emphasises the opportunity and need for custom design tools and compound models that integrate the different digital design models currently available. The proposed framework of an integrated GD system below is one such compound model specific to GD class, which unlike the existing GD systems, allows designers to interact with more than one GD technique at the same time, enabling them to control and modify the design generation mechanisms such that they can potentially explore designs across different design generation paths.

4 Towards the framework of an integrated generative design system

GD systems intended to support design exploration should augment 'reflective practise' (Dearden, 2006; Oxman, 2008; Schmitt & Chen, 1991). GD systems have been used in various innovative ways with many examples of successful implementation. However, existing GD systems lack flexibility in supporting such reflective practice because each system is most often developed based on a single GD technique, constraining the design exploration opportunities to the design generation path offered by the specific GD technique. The proposed framework aims to address this limitation.

A computational implementation of any GD system is a challenging task especially in terms of developing a formal language. This challenge is further exacerbated if a system is to be implemented that integrates multiple GD techniques because such an application would require identification of the common language or translation points across the different GD techniques. The proposed framework of an integrated GD system aims to support multiple GD techniques through:

- Explication of the knowledge used in choice and evaluation of GD techniques to suit the design situation: Human designers are adept at evaluating the design problem such that they are able to choose and formulate the design approach to suit the design needs. This implicit knowledge is used in identifying the basic design elements, operatives and operational mechanisms or the production rules while implementing a GD system. It is suggested that the proposed framework explicates this knowledge such that an integrated GD system can be incrementally developed as an expert system where new rules can be added as identified by the designer, who is also the tool builder.
- Explication of the knowledge needed to formulate, transform or reformulate the problem represented in one GD technique to another, for example, from SG to CA: Not only are the human designers adept at formulating a design problem or choosing a design approach, they are also able to identify the emergent design patterns and reformulate the design problem mid-way through the design development to change the design approach. Once again, the designer's implicit knowledge of this reformulation and transformation can be explicated incrementally as new situations and techniques for transformation are identified and assimilated into the computational system.

Hence, the proposed interactive framework will create a compound model for an integrated GD system. Based on the schematic notation used by Oxman (2006), the proposed framework combines the characteristics of a GD model with that of an evaluation model, Figure 1. There are explicit links between the designer and the evaluation procedure, the evaluation procedure and the

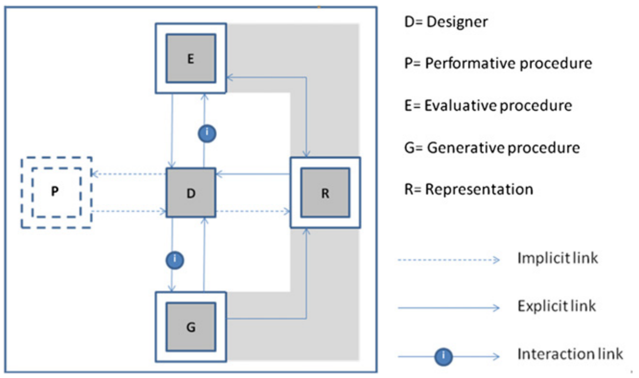

Figure 1 Schematic representation of proposed integrated GD system (based on Oxman, 2006)

representation, and the designer and the generation procedure. The designer builds the knowledge base in the evaluation procedure while the pre-existing knowledge base prompts the designer with a trigger. The explicit link between the evaluation procedure and the representation allows the system to evaluate the emergent design representations using the knowledge base, and this evaluation provides the basis for the trigger prompt. Though the evaluation procedure also proposes potential changes to the generation mechanism it is applied through the designer who explicitly interacts with the generation procedure, which generates the design representation.

The evaluation in the integrated GD system is targeted towards the

- Choice of a GD technique best suited to the design requirements at specific stage of the design development (e.g. level of detail) and the design approach (e.g. bottom-up or top-down);
- Identification of design stages and potential translation points for problem reformulation;
- Identification of requirements and assumptions needed for translation from one formal language to another;
- Providing triggers to the designers based on the incorporated expert knowledge. The evaluation procedure therefore may serve as brainstorming tool for the designer, prompting the designer to explore different GD possibilities emerging during the design exploration. Such an expert system not only allows explication of the designer's implicit knowledge but may also enable the designers to use the explicated implicit knowledge of other designers and experts in the field. Thus, even though the designer remains in control of the design, (s)he can avail some of the benefits of a group brainstorming session through unrestrained triggers for design exploration. This capability will augment another unique characteristic of digital design thinking, which is, explicit bi-directional flow of the information between the designer and the digital medium, which is missing in a paper-based design.

4.1 Theoretical basis for integration through translation points

Though there have been a few examples where more than one of the GD techniques has been used simultaneously for specific purposes, there are no reported efforts of integrating the five different GD techniques together. There is a general agreement that creating a common language is challenging and there is no theoretical proof either that such a common language is possible that allows a generic representation across the different GD techniques. A common language is currently out of scope for this paper and nor does this paper attempt to theoretically establish the possibility of one. Rather, the paper argues towards an approach that may not require a common representation or language and instead identify potential design situations where translation conditions are relatively simpler.

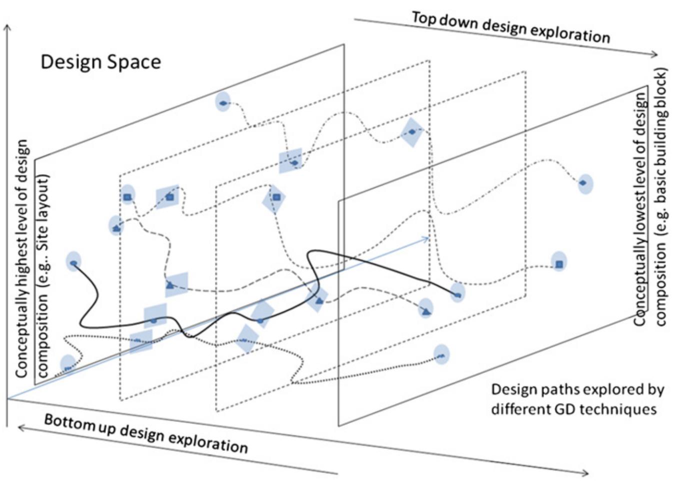
Top down design exploration
Design Space
Conceptually highest level of design composition (e.g.. Site layout)
Conceptually lowest level of design composition (e.g. basic building block)
Design paths explored by different GD techniques
Bottom up design exploration

Figure 2 Schematic representation of design space exploration through different GD paths

Hence, for the validity of the proposed framework it is sufficient if we can conceptually establish that such translation points exist during the design development where switching between different GD techniques and design representations is easier than at other points. For that, let us consider the design space and the potential design solutions explored in this design space through the different GD techniques. For ease of graphical representation, let us visually represent the design space as a three dimensional space, Figure 2. The design space can be explored and navigated across infinite paths depending on how the design emerges, and what GD technique is adopted, e.g., SG path, CA path, etc. The designers may follow a top-down or a bottom-up approach such that they start with decomposing the design problem (e.g. in an urban design scenario, site into zones and sub-zones) or compose solutions from basic design elements and details (e.g. building blocks into sub-zones or sub-zones into zones). In either case GD techniques can be adopted. Though architects and designers typically follow a decomposition re-composition strategy (Akin, 1978; 1994), it is possible to identify different design stages where the design is temporarily finalised at a specific level of detail, for instance, conceptual zoning or site layout, etc. Since these are established landmarks (levels of detail), it is proposed that these design stages provide an explicit stage in the design where the possibility of switchover from one GD technique to another can be evaluated, even though such possibilities for switchover might exist in the intermediate stages as well. Since the design paths explored by each of the GD techniques can be assumed to be non-linear, the solution being explored by any two techniques keep converging or diverging. Hence, at any specific level of detail the evaluation module needs to assess the relative convergence and divergence between the design solutions generated by different techniques.

4.2 Implementation requirements and potential approach

The proposed framework aims to addresses the decomposition/re-composition strategies adopted by designers in the early design phase (Akin, 1978;

1994). Following a bottom-up approach, the designer can start using GD techniques with the smaller design components to achieve designing from part-to-whole. Alternatively, the designer can use GD techniques to follow a top-down approach to achieve designing from whole-to-part. Or the designer can iteratively switch between the bottom-up and top-down approaches, which corresponds more closely to the thinking of human designers who appear to simultaneously work both ways (Lawson, 2005). However, the critical challenge in developing an integrated GD system using the proposed framework is creating the knowledge base for the evaluation procedure. The following types of rules are required for such a purpose, in addition to the production rules within the generation procedure:

- Selection rules: which GD technique is best suited for what problem? These rules also deal with compatibility, i.e., which technique(s) can be used in conjunction for the desired effects. For example (in an urban design scenario), CA is more suited for zoning tasks but using GA with CA can allow optimised zoning. The selection rules can be developed according to the design requirements and characteristics listed in Table 4.
- Trigger rules: These rules are required for identification of potential trigger points or phases in the design development where the designer may consider moving from one GD path to another. In other words, these rules determine the termination conditions of the current generation process. For example (in an urban design scenario), the designer may want a minimum and maximum limit for the number of repeatable elements (e.g., number of blocks) that are clustered together in one module. Once these or similar conditions are met the system should prompt about potential change over opportunity. As the design progresses the design requirements and characteristics may emerge to align closer with one GD technique than the other. The system requires evaluation criteria whereby the proximity of the different GD techniques to the design characteristics can be compared. For example, while working with SG the design may reach a stage where it appears to fall within a structured matrix. At this point of the design there is a potential for a switch to CA. Similarly a design that started with CA may tend to lose the defined cellular structure at the global level. In that case it provides a good indication for switching to some other techniques. These individual triggers and indicators can be implemented as rules which can be administered at regular intervals in the design development as part of the evaluation procedure.
- Translation rules: These rules follow on the Trigger rules to suggest how the identified elements from one GD technique can be translated as inputs to another GD technique. For example, starting with SG when the designer finalises a module, and decides to use it as a part of the cell definition in CA at the next step, what needs to be done? How will a cell or matrix be defined to correspond to the characteristics of the reformulated design problem?
- Strategy rules are required to computationally integrate the bottom-up and top-down approaches. In some ways the Strategy rules integrate the three

rules discussed above, but these rules may be required at a conceptually higher level to organise the design process.

In general, the tables comparing different GD techniques (Section 2) across the technical, design and system development factors provide the initial starting point for developing the different rules for selection, trigger, translation and strategy.

5 *Discussion*

Digital Design thinking is different to traditional paper based design. There is greater potential for interaction, explicit support for reflective practice and bi-directional information exchange. Various classes of digital design models exist with different properties and objectives. GD systems are a class of digital design models that are often intended to support design exploration. However, given that most of the existing GD systems are based on one GD technique they tend to follow one design path rather than exploring different design viewpoints, which is contrary to the design creativity approaches that promote divergent thinking. Hence, compound models of GD systems are needed that can support multiple GD techniques and facilitate more flexible design exploration along different viewpoints. This paper provides one of the first important steps towards this objective and discusses a conceptual framework for such an integrated system.

This paper reviews five different GD techniques to include SG, CA, GA, LS and SI across their technical, design and system development characteristics. Based on this review the paper proposes a framework of an integrated GD system that can support these five techniques, and discusses the challenges and requirements for developing such an system. Developing such integrated computational systems is challenging because of the need for a common language or formalism required for transition across the different GD techniques. The integrated GD system is envisioned as an interactive expert system with the designer being central to the design development. The designer is also viewed as a tool developer, who contributes to the system's knowledge base as (s)he interacts with and uses the system. The system that forms the evaluation procedure requires different categories of rules for problem reformulation, technique selection and translation between the GD techniques. These rules will require explication of some of the knowledge of GD tools and techniques that has so far remained implicit with the tool developers and users.

Some empirical work has been conducted with sample design scenarios to test the feasibility of an integrated framework and identify the conceptual and technical requirements. Future work is needed in refining and implementing the framework. The development of an integrated GD system based on the framework is expected to be an iterative process, much like any other design activity, where the tool builder(s), the designer(s), reflect on the emergent challenges and opportunities as new rules are identified and integrated into the system.

References

Aish, R. (2003). Extensible computational design tools for exploratory architecture. In B. Kolarevic (Ed.), *Architecture in the digital age*. NewYork: Spon Press.

Akin, O. (1978). How do architects design? In J.-C. Latombe (Ed.), *Artificial intelligence and pattern recognition in computer aided design* (pp. 806–809) Amsterdam: North Holland Publishing Company.

Akin, O. (1994). *Psychology of early design*. Pittsburgh: Carnegie Mellon University.

Alfonseca, M., & Ortega, A. (1996). Representation of fractal curves by means of L systems. In *Proceedings of the conference on designing the future* (pp. 13–21). Lancaster: United Kingdom ACM. http://doi.acm.org/10.1145/253341.253348.

Alfonseca, M., & Ortega, A. (2000). *Representation of some cellular automata by means of equivalent L Systems*. In: *Complexity International, 7*.

Altshuller, G. (1984). *Creativity as an exact science*. New York: Gordon & Breach.

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial systems*. Oxford University Press.

Bullock, G. N., Denham, M. J., Parmee, I. C., & Wade, J. G. (1995). Developments in the use of the genetic algorithm in engineering design. *Design Studies, 16*, 507–524.

Caldas, L. G. (2001). An evolution-based generative design system: using adaptation to shape architectural form. Unpublished PhD, Massachusetts Institute of Technology, Boston.

Caldas, L., & Norford, L. K. (2001). Architectural constraints in a generative design system: Interpreting energy consumption levels. In *Building simulation 2001*. International Building Performance Simulation Association.

Carley, K. (1994). Sociology: computational organization theory. *Social Science Computer Review, 12*, 611–624.

Coates, P. S. (2004). Review paper: some experiments using agent modelling at CECA. In *7th generative art conference 2004*.

Coates, P. S., & Carranza, P. M. (2000). Swarm modelling. The use of swarm intelligence to generate architectural form. In *Generative art*.

Coates, P. S., & Hazarika, L. (1999). The use of genetic programming for applications in the field of spatial composition. In *Generative art*. Milan.

Dai, M.-L., & Ozawa, K. (1997). Texture synthesis by L-systems. *Image and Vision Computing, 15*, 197–204.

Dearden, A. (2006). Designing as a conversation with digital materials. *Design Studies, 27*, 399–421.

Deneubourg, J. L. (1977). Application de l'ordre par fulctuations a la description de certaines etapes de la construction du nid chez les termites. *Insect Soc, 24*, 117–130.

Devert, A. (2009). When and why development is needed: generative and developmental systems. In *Proceedings of the 11th annual conference on genetic and evolutionary computation* (pp. 1843–1844). Montreal, Quebec, Canada: ACM. http://doi.acm.org/10.1145/1569901.1570194.

Ding, L., & Gero, J. S. (2001). The emergence of the representation of style in design. *Environment and Planning B: Planning and Design, 28*, 707–731.

Dorst, K., & Cross, N. (2001). Creativity in design process: co-evolution of problem-solution. *Design Studies, 22*, 425–437.

Duarte, J. P. (2005). A discursive grammar for customizing mass housing: the case of Siza's houses at Malagueira. *Automation in Construction, 14*, 265–275.

Frazer, J. H., Frazer, J. M., Liu, X., Tang, M. X., & Janssen, P. (2002). Generative and evolutionary techniques for building envelope design. In *Generative art 2002: 5th international generative art conference GA2002*. Italy, Milan.

Gero, J. S. (1998). Conceptual designing as a sequence of situated acts. In I. F. Smith (Ed.), *Artificial intelligence in structural engineering* (pp. 165−177). Berlin: Springer.

Gero, J. S., & Kazakov, V. (1998). Evolving design genes in space layout problems. *Artificial Intelligence in Engineering, 12*, 163−176.

Gero, J. S., & Kazakov, V. S. (2001). A genetic engineering approach to genetic algorithms. *Evolutionary Computing, 9*, 71−92. http://dx.doi.org/10.1162/10636560151075121.

Halatsch, J., Kunze, A., & Schmitt, G. (2008). Using shape grammars for master planning. In *Design computing and cognition '08* (pp. 655−673).

Herr, C., & Kvan, T. (2005). Using cellular automata to generate high-density building form. In *Computer aided architectural design futures 2005* (pp. 249−258).

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press.

Hybs, I., & Gero, J. S. (1992). An evolutionary process model of design. *Design Studies, 13*, 273−290.

Jackson, H. (2002). *Toward a symbiotic coevolutionary approach to architecture in creative evolutionary systems*. Morgan Kaufmann Publishers Inc. 299−313.

Jacob, C. (1996). Evolving evolution programs: genetic programming and L-systems. In *Proceedings of the first annual Conference on genetic programming* (pp. 107−115). Stanford: California MIT Press.

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

Kelly, G., & McCabe, H. (2006). Interactive generation of cities for real-time applications. In *ACM SIGGRAPH 2006 research posters* (pp. 44). Boston, Massachusetts: ACM.

Knight, T. W. (1998). Designing a shape grammar. In J. S. Gero, & F. Sudweeks (Eds.), *Artificial intelligence in design* (pp. 499−516). The Netherlands: Kluwer Academic Publishers.

Knight, T. W. (1999). Applications in architectural design, and education and practice. In *Report for NSF/MIT workshop on shape computation, NSF/MIT workshop on shape computation*. http://www.shapegrammar.org/education.pdf.

Koning, H., & Eizenberg, J. (1981). The language of the Prairie: Frank Lloyd Wright's Prairie houses. *Environment and Planning, 8*, 295−323.

Krawczyk, R. (2002a). Architectural interpretation of cellular automata. In *Generative art 2002*.

Krawczyk, R. (2002b). Experiments in architectural form generation using cellular automata. In *eCAADe 2002*.

Kunz, J. C., Levitt, R. E., & Jin, Y. (1998). The virtual design team: a computational simulation model of project organizations. *Communications of the Association for Computing Machinery, 41*, 84−92.

Lam, Z., & King, S. A. (2005). Simulating tree growth based on internal and environmental factors. In *Proceedings of the 3rd international conference on Computer graphics and interactive techniques in Australasia and South East Asia* (pp. 99−107). Dunedin, New Zealand: ACM. http://doi.acm.org/10.1145/1101389.1101406.

Lawson, B. (2005). *How designers think, fourth edition: the design process demystified*. Architectural Press.

Lindenmayer, A. (1968). Mathematical models for cellular interaction in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology, 18*, 280–289.

Lindenmayer, A., & Rozenberg, G. (1972). Developmental systems and languages. In *Proceedings of the fourth annual ACM symposium on theory of computing* (pp. 214–221). Denver, Colorado, United States: ACM. http://doi.acm.org/10.1145/800152.804917.

Macy, M., & Willer, R. (2002). From factors to Actors: computational Sociology and agent-based modeling. *Annual Review of Sociology, 28*, 143–166.

Maher, M. L., Smith, G. J., & Gero, J. S. (2003). Design agents in 3D virtual worlds. In R. Sun (Ed.), *IJCAI workshop on cognitive modelling of agents and multi-agent interactions* (pp. 92–100), Acapulco, Mexico.

Mitchell, T. (1997). *Machine Learning*. Mcgraw-Hill International Edit): McGraw-Hill Education (ISE Editions).

Mueller, P., Wonka, P., Haegler, S., Ulmer, A., & Gool, L. V. (2006). Procedural modeling of buildings. *ACM Transactions on Graphics, 25*, 614–623.

Osborn, A. F. (1957). *Applied imagination* (1st ed.). New York: Scribner's.

Oster, G., & Wilson, E. O. (1978). *Caste and ecology in the social insects*. Princeton, NJ: Princeton University Press.

Oxman, R. (2006). Theory and design in the first digital age. *Design Studies, 27*(3), 229–265.

Oxman, R. (2008). Digital architecture as a challenge for design pedagogy: theory, knowledge, models and medium. *Design Studies, 29*, 99–120.

Palubicki, W., Horel, K., Longay, S., Runions, A., Lane, B., & Prusinkiewicz, P. (2009). Self-organizing tree models for image synthesis. In *ACM SIGGRAPH 2009 papers* (pp. 1–10). New Orleans, Louisiana: ACM. http://doi.acm.org/10.1145/1576246.1531364.

Parish, Y. I. H., & Muller, P. (2001). Procedural modeling of cities. In *SIGGRAPH '01: proceedings of the 28th annual conference on computer graphics and interactive techniques* (pp. 301–308).

Payman, A. (2004). *Swarm intelligence*. Pasadena, CA: Jet Propulsion Laboratory.

Poon, J., & Maher, M. L. (1997). Co-evolution and emergence in design. *Artificial Intelligence in Engineering, 11*, 319–327.

Salge, C., Lipski, C., Mahlmann, T., & Mathiak, B. (2008). Using genetically optimised artificial intelligence to improve gameplaying fun for strategical games. In *Proceedings of the 2008 ACM SIGGRAPH symposium on video games* (pp. 7–14). Los Angeles, California: ACM. http://doi.acm.org/10.1145/1401843.1401845.

Savransky, S. D. (2000). *Engineering of creativity, introduction to TRIZ methodology of inventive problem solving*. CRC Press.

Schmitt, G. N., & Chen, C.-C. (1991). Classes of design – classes of methods – classes of tools. *Design Studies, 12*, 246–251.

Schon, D. (1992). Designing as a reflective conversation with the materials of a design situation. *Knowledge- Based systems, 5*, 3–14.

Smith, A. R. (1984). Plants, fractals, and formal languages. In *Proceedings of the 11th annual conference on computer graphics and interactive techniques* (pp. 1–10). ACM. http://doi.acm.org/10.1145/800031.808571.

Speller, T. H., Whitney, D., & Crawley, E. (2007). Using shape grammar to derive cellular automata rule patterns. *Complex Systems, 17*, 79–102.

Stiny, G. (1980). Introduction to shape grammars. *Environment and planning B: Planning and Design, 7*, 343–351.

Stiny, G. (1990). What is a design? *Environment and Planning, 17*, 97–103.

Stiny, G., & Mitchell, W. J. (1978). The Palladian grammar. *Environment and planning* 5–18.

Stiny, G., & Mitchell, W. J. (1980). The grammar of paradise: on the generation of Mughul gardens. *Environment and Planning, 7*, 209–226.

Tapia, M. (1999). A visual implementation of a shape grammar system. *Environment and Planning B: Planning and Design, 26*, 59–73.

Thalmann, D., Hery, C., Lippman, S., Ono, H., Regelous, S., & Sutton, D. (2004). Crowd and group animation. In *ACM SIGGRAPH 2004 course notes* (p. 34). Los Angeles, CA: ACM. http://doi.acm.org/10.1145/1103900.1103934.

Theraulaz, G., & Bonabeau, E. (1995). Modelling the collective building of complex architectures in social insects with lattice swarms. *Journal of Theoretical Biology, 177*, 381–400.

von Neumann, J. (1951). The general and Logical theory of automata. In L. A. Jeffress (Ed.), *Cerebral mechanisms in behavior-the Hixon symposium* (pp. 1–41). New York: John Wiley.

Wolfram, S. (2002). *A New Kind of science*. Wolfram Media Inc.

Yoon, J. S., & Maher, M. L. (2005). A swarm algorithm for wayfinding in dynamic virtual worlds. In *Proceedings of the ACM symposium on virtual reality software and technology* (pp. 113–116). Monterey, CA, USA: ACM. http://doi.acm.org/10.1145/1101616.1101639.