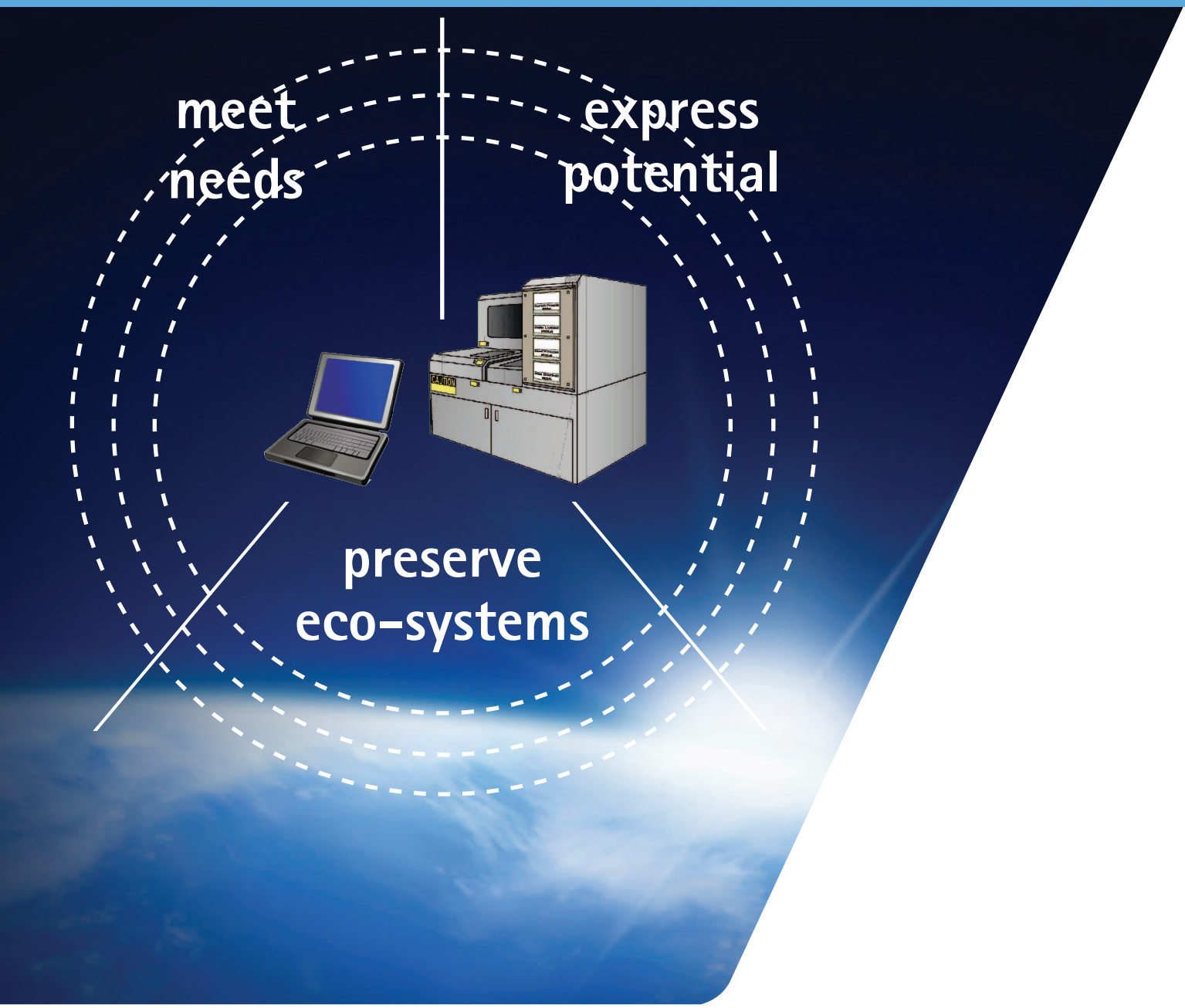
VTT WORKING PAPERS 129



Stephen Fox

Generative Production Systems for sustainable product creation

ISBN 978-951-38-7189-5 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1459-7683 (URL: http://www.vtt.fi/publications/index.jsp)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 5, PL 1000, 02044 VTT
puh. vaihde 020 722 111, faksi 020 722 7001

VTT, Bergsmansvägen 5, PB 1000, 02044 VTT
tel. växel 020 722 111, fax 020 722 7001

VTT Technical Research Centre of Finland, Vuorimiehentie 5, P.O. Box 1000, FI-02044 VTT, Finland
phone internat. +358 20 722 111, fax +358 20 722 7001

Series title, number and
report code of publication

VTT Working Papers 129
VTT-WORK-129

Author(s)
Stephen Fox

Title

Generative production systems for sustainable product creation

Abstract

Increasing the sustainability of human enterprises is a global priority. Sustainability involves individuals, organizations and societies meeting their needs and expressing their potential while preserving natural ecosystems. Advanced manufacturing technologies (AMT) have the potential to enable sustainable product creation. However, the potential of AMT is currently restricted by the limitations of CAD/CAM systems.

Combining AMT with generative computation to develop Generative Production Systems has the potential to overcome the current limitations imposed on AMT by typical CAD/CAM systems. Generative computation automatically produces options that are not stored previously in computer. These options adhere to key requirements, but are unpredictable and involve little, or no, external human input after initial programming. The potential of various types of generative computation to automatically produce designs has been recognized for some years. More recently, it has been proposed that generative computation can be extended from the production of designs to the production of what is described by those designs. This can range from the production of the very large (e.g. buildings) to production of the very small (e.g. micro-electro-mechanical systems).

Findings from the research suggest that implementation challenges can be grouped under the four headings of: inherent ambiguity, domain complexity, computational complexity, and software development. Findings also indicate that there are resources available for meeting these challenges. It is argued that application opportunities for Generative Production Systems can include: implementation of Unified Shape Production Languages which can enable the integration of elicitation with design and production; the introduction of new business models such as Factory 2.0; rapid exploration of materials' potential, rapid generation of new product/component styles; and rapid creation of customer-designed products.

ISBN
978-951-38-7189-5 (URL: http://www.vtt.fi/publications/index.jsp)

Series title and ISSN		Project numbers
VTT Working Papers 1459-7683 (URL: http://www.vtt.fi/publications/index.jsp)		30417; 34632
Date August 2009	Language English	Pages 104 p.
Name of projects Building New Markets; Manufuture	Commissioned by VTT	
Keywords advanced manufacturing technologies, generative computation, generative production systems, unified shape production language, Factory 2.0	Publisher VTT Technical Research Centre of Finland P.O. Box 1000, FI-02044 VTT, Finland Phone internat. +358 20 722 4520 Fax +358 20 722 4374	

Preface

Currently, much of product creation in all economic sectors, from micro-electronics to built environments, is carried out within a paradigm introduced by the industrial revolution. That paradigm is the execution of design, production, and distribution by professional experts working at a few specialized locations. Many business models, ranging from extended enterprises to virtual organizations, have been developed within this established paradigm. Nonetheless, the established paradigm has limited potential to achieve radically improved user-orientation and radically reduced environmental impacts.

These two limitations are extremely significant because, across economic sectors and global regions, organizations face regulatory and market pressures to increase the sustainability of their products and processes. Further, organizations are driven by competitive pressures to differentiate their offerings and increase their sales by increasing the involvement of customers/users. Yet, product users cannot easily participate in the established paradigm. Rather, the established paradigm is populated by professional experts in design and production who find out what product users may want through a variety of professional intermediaries such as market researchers. Moreover, environmental impacts cannot be radically reduced by the established paradigm. For example, production within the established paradigm involves separate parts being manufactured at several different locations, and then being transported long distances for assembly at one or more other locations.

A new paradigm comprising point-of-demand product creation by non-experts could better enable user-orientation and bring about radical reductions in environmental impacts. However, this new paradigm cannot be enabled by established computer-aided design / computer-aided manufacturing (CAD/CAM) systems which are developed for use by professional experts such as building architects, industrial designers, manufacturing engineers etc. By contrast, combining AMT with generative computation may have the potential to enable a wide range customers/users to create the products

that they need/want at point-of-demand. Generative computation automatically produces options that are not stored previously in computer. These options adhere to key requirements, but are unpredictable and involve little, or no, external human input after initial programming. The potential of various types of generative computation to automatically produce designs has been recognized for some years. More recently, it has been proposed that generative computation can be extended from the production of designs to the production of what is described by those designs. This can range from the production of the very large (e.g. buildings) to production of the very small (e.g. micro-electro-mechanical systems). Generative computation combined with AMT can be described as generative production systems. This Working Paper provides an overview of generative production systems and the opportunities which they introduce.

Stephen Fox

Espoo, July 2009

Contents

List of Figures

List of Tables

1. Introduction

1.1 Background

Increasing the sustainability of human enterprises is a global priority (Karl et al., 2009; Pachauri and Reisinger, 2007). As illustrated in Figure 1, sustainability involves individuals, organizations and societies meeting their needs and expressing their potential while preserving natural ecosystems (Buckley et al., 2008).

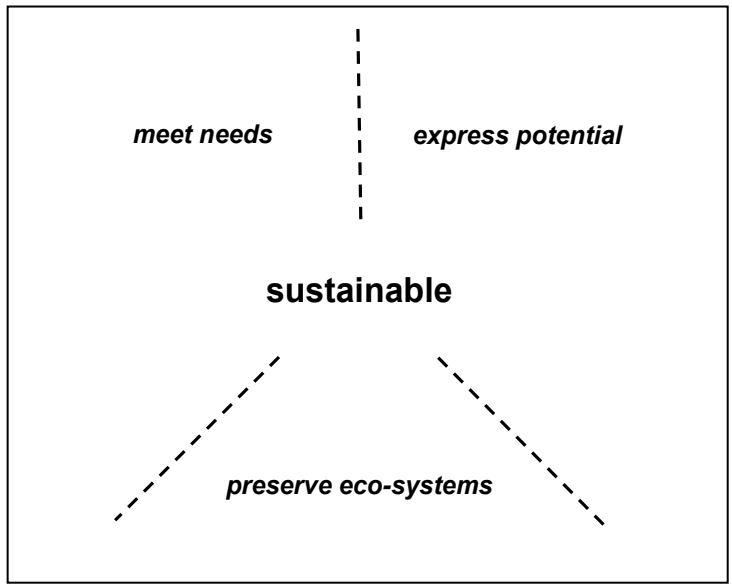


Figure 1. Three criteria of sustainability.

It is explained in the second section of this report how advanced manufacturing technologies (AMT) have the potential to meet all three sustainability criteria during the creation of physical goods. It is also explained in that section, why the potential of advanced manufacturing technologies is currently restricted by the functionality of

established computer-aided design (CAD) and computer-aided manufacturing (CAM) technologies.

In the third section, the potential of generative computation to overcome current CAD/CAM restrictions is explained. As highlighted in Figure 2, generative production systems are defined as generative computation combined with advanced manufacturing technologies. An outline of the foundations of generative computation is also provided. In particular, the importance of formal grammars is discussed.

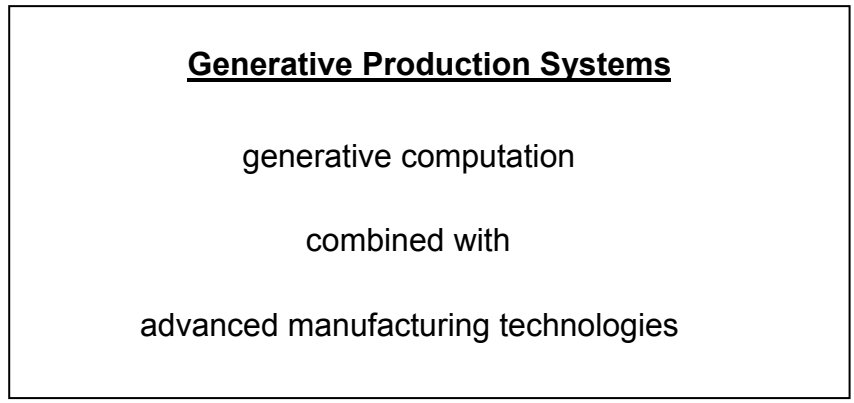


Figure 2. Definition of Generative Production Systems.

In the fourth section, details are provided about the formulation of shape grammars: that is one type of formal grammar which has particular relevance to product creation. The major stages of formulation are described. The stages include: defining a vocabulary of basic forms, developing rules for the manipulation of those basic forms.

In the fifth section, details are provided about the computation of shape grammars. These include: definition with shape algebras, enabling with algorithms, description with pseudo-code, implementation with software, and the combination of shape grammars with other computational methods.

Overall, the analysis of shape grammars provides insights into the formulation and computation of formal grammars in general. Further, the consideration of combination with other computational methods reveals that one formal grammar can provide only a starting point for generative computation, and generative production systems. For example, different types of grammars such as shape grammars and graph grammars can be combined. Further, shape grammars can take input from ontology, and outputs can be improved through a variety of optimally-directed search techniques.

In the sixth section, an overview is provided of challenges in the development of generative production systems. In particular, challenges are described under the headings of: inherent ambiguity, domain complexity, computational complexity, and

software development. Subsequently, potential resources for meeting the challenges are discussed.

In the seventh section, examples are presented of application opportunities for generative production systems. Some of these opportunities are common across economic sectors, while others are particular to certain types of market offerings.

1.2 Research goal

The goal of the research reported in this VTT working paper was to investigate the potential for combination of generative computation with advanced manufacturing technology in order to introduce widespread sustainable product creation. The objectives of the research were to address the following questions: what CAD/CAM limitations can restrict the potential of advanced manufacturing technologies to meet sustainability criteria; to what extent can generative computation overcome those limitations; what are the foundations of generative computation; what are common stages and important details in generative computation; what challenges are there to broader implementation of generative computation, and its combination with advanced manufacturing technologies; what resources are available to meet those challenges; what are the major application opportunities for generative computation combined with advanced manufacturing technologies.

1.3 Research method

The research comprised literature review and exploratory interviews with experts in generative computation, and advanced manufacturing technology.

2. Sustainable product creation

2.1 Overview

Creation means bringing something into existence. Here, product creation means bringing physical products into existence through their design and production. Sustainable means that processes are able to be carried out for an indefinite period in such a way that individuals, organizations, and societies are able to meet their needs, and express their greatest potential in the present, while preserving natural ecosystems. Accordingly, sustainable product creation can be considered under the three headings of: meeting needs, expressing greatest potential, and preserving natural ecosystems. The definition for sustainable product creation used in this paper is presented in Figure 3.

Sustainable Product Creation

design and production of physical products

with technologies that enable individuals, organizations and societies

to meet their needs and express their greatest potential

while preserving natural ecosystems

Figure 3. Definition of sustainable product creation.

In this section, the potential of advanced manufacturing technologies (AMT) to enable sustainable production creation is described. First, an outline of different categories of AMTs is provided. Then, opportunities for AMTs to enable major reductions in the extraction and processing of natural resources are discussed. Next, opportunities are

outlined for AMTs to enable more people to expresses their greatest potential. Subsequently, opportunities for AMTs to meet needs for social products and distributed employment are described. In conclusion, challenges for effective implementation of AMT are outlined.

2.2 Advanced Manufacturing Technologies (AMT)

Advanced Manufacturing Technologies (AMT) enable improved production and/or components by surpassing, in one or more characteristics, traditional combinations of manufacturing and materials. Different types of AMT can be grouped into the following three broad process categories: subtractive processes (e.g. electrical discharge machining, high speed milling, laser processing); forming processes (e.g. direct production casting, incremental sheet forming, reconfigurable molds, robotic bending); and additive processes. It is important to note that additive processes can be divided into non-layer additive processes, which can realize one dimensional complexity (e.g. laser joining, robotic welding), and layer additive processes which can realize three dimensional complexity (e.g. 3D printing, laser sintering, stereolithography, ultrasonic consolidation). Common across AMT is their potential to be digitally-driven by direct transfer of digital design data.

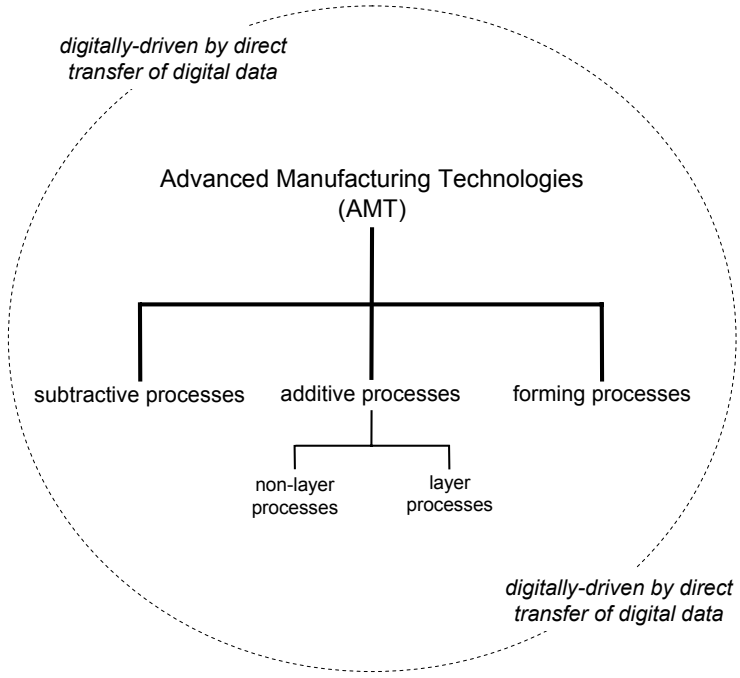


Figure 4. AMT (Advanced Manufacturing Technologies).

AMT can be used to produce everything from molecular sized solid components to very large volumetric products such as entire buildings. Further, AMT makes it possible to transcend previously intractable trade-offs including: performance requirements versus materials microstructures; and geometric complexity versus production time/cost. This enables the introduction of new types of high performance physical goods which are person-specific, location-specific, and/or event-specific.

2.3 Preserving ecosystems

With regard to preserving ecosystems, sustainable product creation should involve reduced consumption of natural resources, which leads to major reductions in the extraction and processing of natural resources which destroys natural ecosystems. Use of Advanced Manufacturing Technologies (AMT) can lead to reduced consumption of natural resources in several ways. First, consumption of raw materials can be reduced by production of topologically optimized designs which have, for example, minimal wall thicknesses. Also, excessive raw material wastage arising from traditional subtractive processes can be eliminated. Further, AMT can radically reduce the need for tooling and, hence, consumption of the raw materials that are traditionally used in the production of tooling. Furthermore, clean lightweight AMT machinery can enable point-of-demand production, and thereby radically reduce the need for packaging materials. A summary is provided in Table 1 below.

Table 1. AMT enablers for reducing consumption of natural resources.

Opportunity	**Enabling factor**
Production Materials	Topologically optimized designs
	More net shape production
	Much less need for tooling
	Packaging of goods for transportation is not required
Process Energy	Fewer energy intensive conversion processes
	Fewer secondary manufacturing and assembly processes
	Refurbishing, repairing, remanufacturing existing components
	Less transportation and transportation of lighter goods

In addition to multiple reductions in raw materials usage, AMT can enable multiple reductions in energy consumption. First, AMT can replace traditional production processes where energy is consumed in heating and cooling cycles, for example, when

converting solid materials into molten liquids. Energy consumption is also reduced by the reductions in secondary manufacturing processes and assembly processes. These reductions are achieved when AMT is applied to produce net shape consolidated assemblies. Also, AMT can reduce transportation energy consumption through point-of-demand production of consolidated assemblies, which traditionally have been manufactured as several separate parts at several different locations. Further transportation energy consumption reductions are possible when goods produced by AMT are lighter in weight because they are topologically optimized. These goods can be lighter weight products which are being transported, and/or lighter weight components which are part of the transportation vehicles themselves (Reeves, 2008). Certain AMT techniques are also well-suited for modernizing, refurbishing or repairing existing components for further use, or remanufacturing them as updated or new products. This continued use of existing components can involve less consumption of raw materials and energy than the production of entirely new components.

2.4 Expressing greatest potential

With regard to expressing greatest potential, sustainable product creation should involve increased scope for self-expression among non-experts in the design and production of physical goods that they need and/or want. It should also involve increased scope for establishing income generating enterprises that provide employment - irrespective of geographical location. There are a number of reasons why AMT has the potential to enable greater self-expression among non-experts. First, AMT can be used to produce goods that comprise only a few consolidated assemblies rather than many separate parts. Hence, only one, or very few, machines are needed to produce goods. As a result, the range of production knowledge required to produce goods is radically reduced. Further, AMT can be used to produce goods at point of demand. This is because the reduced need for specialist heavy equipment and inventory goods obviates the need for traditional factories. Thus, non-experts can stay close to home and need to know much less in order to create products themselves. Further, the production of single products, which are person-specific, location-specific, and/or event-specific, is much more economically viable using AMT because investment is not required in production tooling, etc. Moreover, AMT enables creation of complicated geometries and integrated functionality that can express, for example, one person's unique aesthetic ideas; particular functional requirements; and/or physical characteristics. Unique aesthetic ideas can come wholly from one person's imagination or can involve adaptation of, for example, characters from a computer game which can be downloaded for physical creation using AMT.

There are several reasons why AMT offers increased scope for establishing income generating enterprises that provide employment. First, there is much less need for capital investment because there is reduced need for specialist heavy equipment, inventory goods, and factories for holding them. Further, it can be possible to rent time on AMT machinery that is owned and operated by others. Also, there can be little, if any, need for costly conventional market research. This is because, for example, the goods that are produced can be person-specific and manufactured to order on receipt of order. In addition, there is less, if any, need for costly conventional distribution arrangements. This is because AMT is digitally-driven. Thus, digital product data can be uploaded to the AMT facilities that are nearest to the customer. The potential for digital distribution opens up potential for an enterprise based at one single location anywhere in the world to create and serve a market that spans the whole world. All together these enablers radically reduce the uncertainties that have traditionally caused many entrepreneurial ventures to fail and prevented many potential entrepreneurial ventures from even starting. It is important to note, that established business models are not likely to encompass the sustainability requirement of expressing greatest potential. This is because the established business models, as summarized in Figure 1 above, are the domain of specialized professional experts. The business models of engineer-to-order and design-to-order do offer authority, rather than just choice, to individual customers. However, individual customers express themselves through professional experts such as building architects and industrial designers. Thus, individual customers do not express themselves directly, but do so through intermediaries. Although it is clear that the creation of very large physical goods, such as cruise ships, will continue to involve the participation of hundreds of people, AMT has the potential to enable individuals to express their needs and wants much more directly through the rapid creation of physical scale models. A summary of AMT enablers for increased self-expression is shown below in Table 2.

Table 2. AMT enablers for expression of potential.

Opportunity	**Enabling factor**
Creation of goods	Less production expertise needed
	Production can be carried out at a single location
	Production of person-specific goods is economically viable
	Possible to produce from external sources such as computer games
Establishing enterprises	Little, if any, initial capital investment in production plant
	Conventional market research is not necessary
	Digital, rather than physical, distribution
	Can create and serve a global market from just one location

2.5 Meeting needs

With regard to meeting needs, sustainable product creation involves meeting ever increasing demand for social products such as medical aids and low-cost housing. AMT can enable the creation of a wide variety of person-specific medical goods from scan data, including: conformal seating; crash helmets; dental aligners; dental bridges and crowns; hearing aids; orthotic footwear; prosthetics; surgical cutting guides; surgical implants. Person-specific medical goods can provide better performance. Person-specific hearing aids, for example, can provide increased comfort and increased performance through reduced feedback. Different people can make different distinctions between needs and wants. One person, for example, may regard perfect alignment of their teeth as being a need. By contrast, another person may regard perfect alignment of teeth as being a want. In either case, person-specific transparent dental aligners, which are more attractive and less invasive than traditional dental braces, can be produced with AMT. Similarly, one person may need a dental crown and want it to made from gold. By contrast, another person may need a dental crown and be satisfied to have it made from a non-precious material. In either case, more accurate and less expensive dental bridges and crowns can be made directly from patient scan data using AMT.

In addition to medical goods, AMT has the potential to revolutionize the creation of low cost housing. This can be achieved through the digitally-driven manufacture of interlocking envelope components at point-of-demand. Further, these components can be assembled right first time by non-experts. Moreover, AMT has the potential to enable improved performance from all building types through the embedding of sensors, actuators etc., into building components during their manufacture. Embedded devices can be used to make buildings responsive to, for example, heat loss through open windows by triggering the closing of the open window or the sending of a message for a person to do so. In addition, AMT has the potential to enable improved performance from all building types through the more rapid and economic production of location-specific devices for capturing and conducting heat. These devices could have, for example, the specific and complex geometries required to most efficiently capture solar energy at a particular inner city rooftop.

Importantly, AMT has the potential to meet the need for distributed employment throughout nation's regions. Within existing business models, product development, product production and product despatch are often concentrated at the few physical locations of companies' premises. The number of physical locations can become even fewer when companies off-shore product development, product production, and/or product despatch to other countries to take advantage of lower resource costs. The concentration of employment in a few locations leads to nations having regions of disproportionately high under employment and/or unemployment. As a result, nations can have regions of under population with consequent national problems such as

infrastructure being under-utilized; long-term territorial integrity being compromised; etc. (e.g. Beale, 2000). By contrast, AMT can enable production, and hence employment, at any location. This is because of the enablers outlined above: reduced investment in plant; reduced need for costly market research; digital, rather than physical distribution; and potential to serve a global market from one location. Further, the labour cost component of product created with AMT is relatively low. This factor, combined with the potential to digitally distribute production to each particular point-of-demand, means that there is little incentive to outsource from a higher labour cost economy to a lower labour cost economy. As well as it being technically feasible and economically viable to produce goods at point-of-demand, it is also commercially beneficial. This is because meeting increasing person-specific, location-specific, and/or event-specific demand through concentrated product development, production and despatch introduces increasing complexity into existing business models. Thus, point-of-demand production can reduce non-value adding costs, such as production defects and overtime working, that arise from complexity. A summary of AMT enablers for meeting needs of individuals, organizations, and societies is shown in Table 3.

Table 3. AMT enablers for meeting needs of individuals, organizations and societies.

Opportunity	**Enabling factor**
Social products	More economic production of higher performance medical goods
	More economic production of higher performance buildings
Distributed employment	Feasible and viable to manufacture anywhere
	Commercially beneficial to manufacture anywhere

2.6 Challenges for effective implementation of AMT

As illustrated in Figure 2, the overview provided in the preceding paragraphs describes the potential of Advanced Manufacturing Technologies (AMT) to address all three sustainability criteria: meeting needs; expressing greatest potential; and preserving natural ecosystems.

However, design for AMT production processes presents a number of challenges for effective implementation of AMT. Firstly, AMT introduces many new design spaces for designers to explore. These new design spaces arise from the capabilities of AMT to produce, for example, complicated geometries. As outlined in the preceding section, AMT can enable the production of topologically optimized designs which have, for example, minimal structural thicknesses. This can involve the production of

complicated constructions such as lattices or honeycombs. Moreover, it can involve discontinuous (i.e. digital) material placement rather than the continuous (i.e. analogue) material placement that is typically used in manufacturing. Further, AMT can enable the consolidation of many parts into single piece assemblies. These assemblies will often have much greater geometric complexity than the individual parts which they supersede. As well as complex geometries, which are "inside" products, AMT can enable the production of complex geometries, which make up the external form of products, such as the envelopes of buildings; the shells of mobile phones, and so on. Designers are largely unfamiliar with the nature of the new design spaces introduced by AMT. Further, the scope of design spaces opened up by Advanced Manufacturing Technologies (AMT) is so vast, and expanding so rapidly, that they cannot be explored quickly or comprehensively by human designers using only traditional CAD tools. Accordingly, new computational methods are required to enable definition and exploration of AMT-enabled design spaces. Some of the outcomes could be, for example, definition of geometric ranges for sustainable low-cost social housing made from alternative types of indigenous materials; definition of geometric ranges for mobile phones within an established brand identity.

A second challenge for widespread adoption of AMT arising from design is the difficulty of modelling geometrically complex constructions using current CAD and CAM systems. These difficulties are aggravated when a single component comprises multiple materials. In simple terms, current CAD and CAM systems become slower and less reliable the more surfaces and materials have to be encompassed within computation. Moreover, current CAD and CAM systems are typically dependent on going human input during their operation. During the exploration of design spaces, for example, they rely on continual decision making by human designers. As a result, the exploration of design spaces is extremely time-consuming. Further, it is likely to be influenced by designers' prior training and experience of, for example, established design for manufacture and assembly principles which are made obsolete by AMT. Accordingly, new computational methods are required to enable rapid and reliable product creation using AMT.

A third challenge is diversity of people who could make good direct use of AMT. Traditionally, the use of sophisticated design/production equipment has been restricted to professional experts who have extensive training and supporting colleagues. However, AMT has the potential to enable production of person-specific, location-specific, and/or event-specific physical goods close to point-of-demand by people who do not have to be experts in AMT. These non-experts could include, for example, medical professionals such as surgeons and dentists; small business operators such as furniture makers and boiler makers; enthusiasts for hobbies such as hunting, shooting and fishing. A boilermaker, for example, carrying out refurbishment of a plant room

could use AMT to produce a single assembly instead of having to cut, bend and join several parts. Such use of AMT could be enabled by having proprietary AMT "stations" within all the branches of an industrial wholesaler. Similarly, person-specific accessories for hunting, shooting and fishing equipment could be enabled by also having interactive, easy-to-use AMT "stations" within the outlets of an outdoor goods retailer. However, the drawback of current CAD and CAM systems is that, as well as being incompatible with AMT capabilities, they are not easy-to-use without extensive prior training. To draw an analogy, what is required is AMT "stations" that are similar to, for example, check-in automats at airports: rather than the check-in software systems which are used by trained airport ground crew behind check-in desks. This will involve establishing computer-enabled design procedures for AMT that are much simpler, but can still meet inherent domain requirements for safety etc., as well as large sets of problem specific requirements and constraints. A summary of challenges to effective implementation of AMT arising from design issues is provided in Table 4.

Table 4. Challenges for effective implementation of AMT.

Challenge	**Example**
Vast new design spaces	Topologically optimized structure within complex external form
CAD/CAM limitations	Modelling of multi-surface, multi-material assemblies
Diversity of potential users	Boiler makers; medical professionals

Overall, these challenges mean that the opportunities for sustainable product creation cannot be met by further implementation of established manufacturing materials and machines, conventional CAD software, established human design / engineering skills. The potential of Generative Production Systems to fulfil these challenges is discussed in the next section.

3. Generative Production Systems

3.1 Overview

The potential of various types of generative computation to automatically produce designs has been recognized for some years. More recently, it has been proposed that generative computation can be extended from the production of designs to the production of the artefacts that are described in those designs (Fischer et al., 2005; Hornby, 2005; Hornby and Pollack, 2001). This involves digital production in the virtual reality of CAD and digital production in the physical reality of CAM. For example, digital chair designs are generated through computation and then physically "printed" directly from digital data (CAD/CAM News, 2009). This type of generative design and production may be better enabled by the use of physical voxels in production which match virtual voxels that are used in design (Hiller and Lipson, 2009). The term, voxel, is an abbreviation for "volume element", and can be thought of as being something like a three dimensional version of a pixel. In this paper, the term generative production system is used to encompass manufacture and assembly, as well as design. Moreover, in the penultimate section of this paper, it is proposed that Generative Production Systems could, and should, encompass the elicitation of requirements. In particular, it is argued that Generative Production Systems could make it possible to have a seamless progression from mental visualization of an artefact to physical production of that artefact (i.e. from mind to machine). In particular, the term generative production system is used in this paper to mean generative computation combined with AMT. This definition is highlighted in Figure 3 below.

In this section, important aspects of Generative Production Systems are described. First, production system formalisms are outlined as the underlying mechanism for Generative Production Systems. Then, the use of transformational-generative grammars within production system formalisms is discussed. Next, the potential of one type of transformational-generative grammar to meet challenges for effective implementation of AMTs is described. That particular type of formal grammar is shape grammar. The

computation of shape grammar is described in the fifth sub-section. The capability of shape grammars to enable infinite spatial emergence is discussed in the sixth sub-section. In the concluding sub-section, reasons are given for making reference to shape grammars in the development of Generative Production Systems for sustainable product creation.

3.2 Production system formalisms

Production system formalisms (Post, 1943) can provide an underlying mechanism for Generative Production Systems. Formalisms involve the representation formal structure rather than the representation of content. Typically, a formal system consists of a formal language (e.g. finite strings of letters or symbols) together with a deductive system of inference rules. Production system formalisms are computer programs consisting of a knowledge base of rules and general facts, a working memory of facts concerning the current case, and an inference engine for manipulating both.

Within production system formalisms, rules consist of a condition part (e.g. an IF statement) and an action part (e.g. a THEN statement). With the condition part of a rule being on its left-hand side (LHS) and the action part being on its right-hand side (RHS). Such rules can be described as condition-action rules (i.e. IF condition THEN action), and can be expressed as $A \rightarrow B$. The inference engine in a production system must determine which rules are relevant and choose which one(s) to apply. The inference engine can be described as a *finite state machine* with a cycle consisting of three action states: *match* rules, *select* rules, and *execute* rules. In the first state, match rules, the inference engine tests the conditions (IF statement on LHS) against the working memory. If a rule's precondition matches the working memory, then the production is said to be *triggered*. In the second state, select rules, the inference engine applies some selection strategy to determine which rules will actually be executed. In the third state, execute rules, the inference engine executes the actions (THEN statement on RHS) of the selected rules. When a rule's action is executed, it is said to have *fired*. Usually, the actions of a rule change the working memory, but they may also trigger further processing outside of the inference engine (for example, interacting with users through a graphical user interface or calling local or remote programs). The inference engine then cycles back to the first state and is ready to start over again.

The inference engine stops either on a given number of cycles, controlled by the operator, or on a quiescent state of the data store when no rules match the data. Since the working memory is usually updated by executing rules, a different set of rules will match during the next cycle after these actions are performed. An important element of inference engines is the rule *interpreter*. This must provide a mechanism for prioritizing

productions when more than one is triggered. The three action states (i.e. recognize-act cycle) are summarized below in Table 5.

Table 5. Action states for inference engines in production systems.

State	Action
Match rules	The inference engine tests the condition parts (IF statements on LHS) of rules against the working memory.
Select rules	The inference engine applies some selection strategy to determine which rules will be actually be executed.
Execute rules	The inference engine executes the action parts (THEN statements on RHS) of rules and thus changes the working memory.

3.3 Transformational-generative grammars

As stated above, production system formalisms are computer programs consisting of a knowledge base of rules and general facts, a working memory of facts concerning the current case, and an inference engine for manipulating both. The rules are defined within formal grammars. These are sets of formation rules that describe which strings formed from the alphabet of a formal language are syntactically valid within the language. Formal grammars are congruent with the doctrine of formalism that formal structure, rather than content, is what should be represented. In particular, a formal grammar addresses the location and manipulation of the strings of the language.

Transformational-generative grammars were introduced by Noam Chomsky (1957) in his efforts to develop a mechanism that would generate exactly the set of "grammatical" English sentences. Subsequently, other types of grammars have been developed that generate arrays, trees, graphs, and shapes (Stiny and Gips, 1980). Chomsky developed the idea that each sentence in a language has two levels of representation — a deep structure and a surface structure. He argued that the deep structure is mapped on to the surface structure via transformations involving phrase-structure rewrite rules. In particular, phrase-structure rewrite rules specify that a given phrase may be replaced by another given phrase. The key elements of this conceptualization are summarized by the term, *transformational-generative grammar*. As well as being generative, such grammars can be used to determine what is, and is not, grammatical (i.e. what does, and does not, belong in the language). In Chomsky's grammars, complex forms are created by successively replacing parts of a simple object by using the rewriting rules. It is important to note that rules are applied in sequence (e.g. one at a time) in Chomskyan

grammars. By contrast, rules are applied in parallel (e.g. all at once) in, for example, Lindenmayer grammars (Lindenmayer, 1968).

Chomsky's grammars are defined over an alphabet of symbols, and map strings of symbols into strings of symbols to generate a language of symbols. Shape grammars are a geometrical adaptation of Chomsky's grammars (Speller et al., 2007) in which shape grammar is defined over a set of shapes, and maps shapes into shapes to generate a language of shapes (Stiny, 1976). Despite their similarities, shape grammars differ from Chomsky's transformational-generative grammars in at least two respects. Firstly, symbols that are rewritten are the same symbols that occur in the final design. Secondly, the symbols that are rewritten are geometrical entities, for example line segments, rather than discrete symbols that represent such entities.

Rules within Chomsky's phrase-structure rewrite grammars can be expressed as $A \rightarrow B\ C$. This means that the constituent A is separated into the two sub-constituents B and C. An example is $S \rightarrow NP\ VP$, which means a sentence consists of a noun phrase followed by a verb phrase. Rules within shape grammars can be expressed as $a \rightarrow b$, where both a and b are shapes. Each rule application involves selection of rules; identification of sub shapes; implementation of rule; and generation of new shape. The generation of new shapes through rule implementation can involve shape addition and shape subtraction (Stiny, 1980; 2006). Shape generation can take place in phases: with the level of the solution being lowered from high-level/abstract to a low-level/complete, until it satisfactorily represents the requirements (Deak et al., 2006). In simple terms, rules are chosen from a rule set and applied to an evolving entity to change it in some way (Agarwal and Cagan, 2000).

Through computation, shape grammars have been used to generate designs and the physical artefacts that are described in those designs. Although there are unresolved challenges in the implementation of shape grammars, they may provide the most comprehensive and best documented source for informing the development of Generative Production Systems. This is because the formulation and computation of shape grammars has been carried out by range of scientists, and reported in refereed scientific periodicals, for more than three decades. The following sub-sections provide examples of shape grammar applications; an overview of shape grammar computation; and a discussion of how infinite spatial emergence is enabled by shape grammar computation. Subsequently, the potential of shape grammars to inform development of Generative Production Systems for sustainable product creation is considered.

3.4 Shape grammar applications

Shape grammars have been formulated within research for a variety of product types including: automobiles (Orsborn et al., 2006); bicycles (Suppapitnarm et al., 2004); buildings (Stiny and Mitchell, 1978); cameras (Lee and Tang, 2006); chairs (Hsiao and Chen, 1997); coffeemakers (Agarwal and Cagan, 1998); mobile telephones (Ahmad and Chase, 2006); motorcycles (Pugliese and Cagan, 2002); transmission towers (Shea and Smith, 1999). Within research, shape grammars have also been formulated to generate design options for a variety of component types including: car panels (McCormack, 2002); micro-electromechanical resonators (Agarwal et al., 2000); packaging (Chen et al., 2004); and trusses (Shea and Cagan, 1999). Importantly, novel components designs can be compatible with other components. Novel car body panels, for example, can be compatible with car engines (McCormack and Cagan, 2002).

Also within research, shape grammar rules have been formulated which relate to physical materials and production machinery, as well as to the geometric forms of shapes (Heisserman and Woodbury, 1993; Brown et al., 1995; Wang and Duarte, 2002). Such shape grammar rules enable the manufacture of designs generated by shape grammars (Sass, 2008). Furthermore, shape grammars can enable designs to be produced in different sizes using different types of manufacturing equipment - from the same one file (Sass, 2007a). This offers the possibility of production of a scale model for the purpose of assembly instruction; and production of full-sized components for assembly into a completed product by individuals who do not have relevant prior assembly experience. Also, components can be produced which have accurate friction-fit/snap-fit joints with the necessary parameters for tolerance, material thickness and structural modulation being included into the members of the shape vocabulary (Cardoso and Sass, 2008). This further reduces the amount of assembly skill required.

If shape grammars are to facilitate production, then the shape grammar rules need to be congruent with the properties of materials and the functionality of machines (Sass and Oxman, 2006). Many board materials, for example, are supplied in flat sheets with a limited number of standard stock sizes. Machines for milling sheet materials often have flat beds and have, so called, two and a half D milling paths (i.e. 2D plus thickness). Accordingly, if board materials are to be used in production, shape grammar rules need to be congruent with their properties (e.g. standard stock sizes) and associated machining techniques (e.g. flat bed milling). Generation of machine information from design information can be a three-stage process if a product is to be assembled from a number of components. First, a design description is generated. Second, definitions of material properties need to be applied in order to determine sub-assembly dimensions etc. Third, definitions of machine functionality need to be applied in order to define tool paths etc (Sass and Oxman, 2006). Shape grammars rules have been formulated which relate to a range of production processes including: fuse deposition modelling (Wang

and Duarte, 2002); laser cutting, plasma cutting, and water-jet cutting (Kilian, 2003; Sass et al., 2005); sheet notching, bending and punching (Soman et al., 2003); and stereo-lithography (Heisserman and Woodbury, 1993).
In addition, within research, it has been demonstrated that expressions can be associated with shape grammars rules which enable appraisal of manufacturing cost throughout the design process (Agarwal et al., 1999). However, most applications of shape grammars have focused on the development of product forms rather than the satisfaction of functional requirements (Ang et al., 2006). Nonetheless, it has been argued that shape grammars can be particularly useful for products that are differentiated primarily on the basis of form but are driven by function (Agarwal et al., 1999). As summarized in Table 6 below, shape grammars can enable digitally-driven design, manufacture, assembly.

Table 6. Applications of shape grammars within research.

Application	**Benefit**
Design	Multiple design options can be generated, which do adhere to defined constraints, but involve little, or no, input by the user.
Manufacture	Components can be produced in different sizes using different types of manufacturing equipment – from the same file.
Assembly	Scale models and full-sized components can be produced to communicate and facilitate assembly by non-experts.

Importantly, it is possible for these digitally-driven processes to be carried out by non-experts. Moreover, automatic generation of emergent geometries is fundamental to computation of shape grammars. Thus, the computation of shape grammars has clear potential for addressing the challenges in successful implementation of AMT summarized in Table 6 above: new design spaces; CAD/CAM limitations; and diversity of potential users.

3.5 Shape grammar computation

A sequence of shapes is generated during the computation of shape grammars. Each shape, except for the first shape, is generated automatically from the previous shape by the application of one or more rules (Knight, 2003a). The definition and application of rules involves algebras (e.g. Krstic, 2001). In particular, shape algebras (Stiny, 1991) within which letters or other symbols are used to represent shapes rather than numbers.

Formally, a shape grammar has four components: S is a finite set of shapes; L is a finite set of symbols; R is a finite set of shape rules in the form $a \rightarrow b$; and I is an initial

shape. Applying a shape rule involves the following three steps: First, compare the left shape under one or more Euclidean transformations (e.g. reflection, rotation, scaling, translation). Second, IF there is a match, THEN subtract the left shape under the same transformation and, third, add the right shape under the same transformation. These steps can be expressed more precisely as: IF $t(a) \leq c$, THEN $c' := [c - t(a)] + t(b)$. a being the left-hand shape; b being the right-hand shape; c being the current shape; c' being the new current shape; t being one of the Euclidean transformations or a finite sequence thereof; - meaning subtract or erase; + meaning add or draw; $\leq$ meaning is a part of; := meaning is assigned the value of. If there is more than one transformation under which the left shape matches the current shape, then there is more than one way to apply the rule. Parametric grammars are extensions of regular grammars, and the steps in their computation can be expressed as: IF $t(g(a)) \leq c$, THEN $c' := [c - t(g(a))] + t(g(b))$. With g being a parametric assignment subject to some constraint (Li, 2002; Stiny, 1991). All shape generation must start with an *initial shape* such as a point, or a coordinate axis. This initial shape need not be visible in the final design which is generated. It can be, for example, a global or a local coordinate system (Orsborn et al., 2006). If shape generation is to end, there should be a *terminal rule* that prevents any other rules from being applied after it. Alternatively, shape generation can continue indefinitely and outputs could be chosen at any point in the process.

The computation of shape grammars can be driven by "hard coding". This involves attaching alphanumeric labels and symbol markers to a shape. The labels and markers determine which rules can and cannot be applied to the shape. As a result, they ensure that mutually exclusive rules cannot be applied to the same design. Labels that describe function, for example, can be used to maintain the proper function-to-form sequence, and to ensure that all functional requirements are met before the generation process terminates. Also, certain elements of design become important only under certain conditions and labels can be used to indicate the applicability of rules (Agarwal and Cagan, 2000). Properties can be included with the inclusion of, so called, weights (Stiny, 1992). Properties can be aesthetic, functional, structural, and so on. For example, points can have diameters; lines can have thicknesses; planes can have colours; solids can have materials; and so on. In a study involving generation of micro-electromechanical systems (MEMS), the four main elements (central mass, actuators, anchors, and springs) were each assigned a different weight to ensure that feasible designs were generated. In particular, shapes of higher weight overlapped and erased shapes of lower weight (Agarwal et al., 2000).

Also, the computation of shape grammars can be driven by parametric shape recognition. This enables any shape within predefined allowable limits to be recognized by the rule and thus allows the rule to be applied. Parametric shape recognition offers more possibilities for emergent design than only hard coding. It also offers possibilities

for faster implementation than hard coded shape grammars. However, parametric shape recognition is more challenging than hard coding, because a rule needs to recognize a shape regardless of scaling, rotation and/or other allowable transformations (McCormack and Cagan, 2002; 2006). Hard coding and shape recognition can be combined (Orsborn et al., 2006). A summary of important enablers in shape computation is provided in Table 7 below.

Table 7. Important enablers in shape computation.

Enablers	**Example**
Initial shape	Coordinate system
Terminal rule	Removes labels that enable rule application
Generation in phases	High-level to low-level
Hard coding	Labels, markers, weights
Parametric shape recognition	Recognizes a shape regardless of scaling, rotation etc.
Enabling software	Shape grammar interpreter

As stated above, shape grammars are examples of production system formalisms. That is computer programs which include rule interpreters within inference engines. Accordingly, a computer program for shape grammar generation can be referred to as a shape grammar interpreter (Heisserman, 1994; Krishnamurti, 1982; Krishnamuriti and Giraud, 1986; Piazzalunga and Fitzhorn, 1998; Tapia, 1996, 1999). The user can guide the program in selecting the rule to be applied and where in the current shape to apply it. Alternatively, an automated optimization routine can make the choices presented by the interpreter concerning rule selection, application, and parameters during instantiation. After each rule application, the design is examined, and if the design is deemed completed, it is presented to the user. If not, the design is modified by the user or by the optimization routine (Cagan et al., 2005). Perhaps the most sophisticated type of shape grammar interpreter is one that uses parametric subshape recognition. Such an interpreter surpasses shape grammar interpreters that are limited to nonparametric subshape matching or parametric shape matching (McCormack and Cagan, 2006). It has been argued that a program suitable for carrying out analyses using shape grammars can be referred to as a parsing program. This type of program would be given a shape grammar and a shape. The program would determine if the shape is compatible with the grammar and, if so, gives the sequence of rules that produces the shape. A third type of program could generate a shape grammar based on automated analysis of a corpus of existing designs. Such software could be called an inference program. A fourth type of

program would be a CAD program that would help the user design shape grammars (Gips, 1999).

3.6 Infinite spatial emergence

The operation of shape grammars should emulate what people do when they draw or erase shapes, build, modify or move shapes around producing different spatial relations (Krstic, 2008). As stated above, shape grammars can be considered to be a geometrical adaptation of Chomsky's grammars (Speller et al., 2007). Chomsky theorized that unlimited extension of a language such as English is possible only by the recursive device of embedding sentences in sentences. The term, recursion, is used in mathematics and computer science to refer to a class of objects or methods defined by a simple base case (or cases) and rules to reduce all other cases toward the base case. For example, the following is a recursive definition of a person's ancestors: one's parents are one's ancestors (base case); the parents of one's ancestors are also one's ancestors (recursion step). In other words, the output of the rule may be used as an input to the same rule either immediately or after other operations have occurred. Thus, recursion is the process a procedure goes through when one of the steps of the procedure involves rerunning the procedure. A procedure is a set of steps that are to be taken based on a set of rules. The running of a procedure involves actually following the rules and performing the steps. An analogy might be that a procedure is like a menu in that it is the possible steps, while running a procedure is actually choosing the courses for the meal from the menu. Recursion can enable definition of an infinite set of objects by a finite statement. Accordingly, an infinite number of computations can be described by a finite recursive program, even if this program contains no explicit repetitions (Wirth, 1976). The computation of a shape grammar involves a recursive sequence of rule applications.

Within shape grammars, a *shape* is a finite arrangement of spatial elements from among points, lines, planes, volumes, or higher dimensional hyperplanes, of limited but non-zero measure. Within shape grammars, shapes can be constructed from certain parts and then decomposed into other parts that become the basis for continuing the computation (Knight, 2003b). Any of the subshapes (parts) of a shape can be recognized and used in a shape computation. Points, lines and planes that are parts of shapes, for example, can be recognized and transformed through the execution of shape rules (McCormack and Cagan, 2002). In other words, shapes do not have fixed primitives that limit the ways in which they can be computed. Rather, shapes are moving targets. Thus, shapes are ambiguous throughout computation and can vary as computations unfold (Stiny, 1999). Moreover, the emergent shapes that can be recognized in any step of a shape grammar computation can be limitless in number because shapes are freely

decomposable (Knight, 2003a). This is very different from the traditional shape representations used in computer graphics systems. There, shapes are often represented as sets of lines that cannot be further decomposed, i.e. no portions of lines can be easily recognized. In Constructive solid geometry (CSG), for example, primitives are cuboids, cylinders, prisms, pyramids, spheres and cones. The set of allowable primitives being limited by each software package.

Shapes can be within a Cartesian coordinate system (Stiny and Mitchell, 1978). That being a system for locating and measuring points in space based on a three dimensional axis labeled X, Y, Z. Shapes can be parametric. That is, shapes may have no intrinsic structure. They can be indefinite with some details or characteristics that are fixed and others that vary. The characteristics that vary are the variables of the shape. The range of variation for each variable can be set by conditions that values assigned to the variable must satisfy. When values are assigned to the variables of a parametric shape, a definite shape is defined.

A shape is part of another shape if it is embedded in the other shape as a smaller or equal element. A part of a shape can be called a *subshape* (Stouffs and Krishnamurti, 1994). The form of a bottle, for example, has been decomposed into subshapes such as: cap, upper part, label region, lower part, and bottom (Chau et al., 2004). The form of a mobile phone, for example, has been decomposed into body and fascia (Ahmad and Chase, 2006). The form of motor vehicles, for example, has been decomposed into front wheels, rear wheels, front wheel well, rear wheel well, front fender, rear fender, front bumper, rear bumper, front windshield, rear windshield, grill, headlight, hood, roof, trunk, taillight, rocker, door, front side window, rear side window, door handle, ground, belt line (Orsborn et al., 2006). Then, for example, headlight can be decomposed into curves, and hood can be decomposed into straight lines (McCormack and Cagan, 2002; 2006). Within research, for example, the complicated forms of a sample of forty-two different motor vehicles were captured using the basic form of four-control-point Bezier curves (Orsborn et al., 2006). The connectivity of subshapes can be maintained by allocating labels to them. It is important to note that there can be alternative decompositions of the same shape depending on the domain of analysis. Shapes can be categories within families of shapes that, for example, comprise a particular assembly within a product. In addition to shapes and subshapes that are sufficiently regular to be described easily in traditional Euclidean geometry, shapes can include fractals. Those are rough or fragmented geometric shapes that can be split into parts, each of which is approximately a reduced-size copy of the whole. Natural objects that approximate fractals to a degree include clouds, mountain ranges, lightning bolts, coastlines, and snow flakes. In shape grammars for fractals, the number of rules is small and the number of recursions high (Ediz and Çağdaş, 2007; Schmitt and Chen, 1991). Important characteristics of shapes within shape grammars are summarized in Table 8.

Table 8. Important characteristics of shape within shape grammars.

Characteristic	**Summary**
Decomposable	Shapes can be decomposed into subshapes that can be described easily in Euclidean geometry or as fractals.
Ambiguous	Shapes do not have fixed primitives that limit the ways in which they can be computed. Rather, shapes can vary as computations unfold.
Parametric	Shapes can be indefinite with some details or characteristics that are fixed and some that vary.
Maximal	A maximal spatial element cannot be combined with any other spatial element in the same shape to form a larger spatial element.

Overall, shape decompositions are sets of shapes that emphasize certain properties of their sums (Krstic, 2008). The structure of decomposition may be seen on two levels: local and global. On the local level the decomposition is a set of shapes which puts emphasis on the relations among its elements. By contrast, on the global level the decomposition is seen as analyzing a shape (the sum of its elements) so that the relations between parts of the shape and elements of the decomposition are exposed. If a decomposition is to be an approximation of a shape, then its structure on a global level is of the most importance. A decomposition that serves as a shape approximation should have the unique representation for each part of the shape it analyzes (Krstic, 2008). Within shape grammars, an important enabler of shape and subshape recognition is maximal representation. A spatial element in a shape is denoted a maximal spatial element if it cannot be combined with any other spatial element in the same shape to form a larger spatial element. If a shape only contains maximal spatial elements, the shape is termed *maximal* and its representation is a *maximal representation* (Krishnamurti, 1992; Stouffs and Krishnamurti, 1994).

As highlighted in Figure 5 below, the recursive application of transformation rules to shapes that are decomposable, ambiguous, parametric and maximal makes if possible for one shape grammar to produce an infinite number of options for design and/or production (Orsborn et al., 2006). Moreover, the way that shape is handled in shape grammars enables the options to be emergent rather than predictable.

Infinite Spatial Emergence

recursive application of transformation rules

to

decomposable, ambiguous, parametric, maximal spatial elements

Figure 5. Mechanism for infinite spatial emergence.

The emergent shapes that can be recognized in any step of a shape grammar computation can be limitless in number because shapes are freely decomposable into subshapes such as points, lines and/or planes, rather than just shapes such as circles and squares. An emergent shape can be the sum of shapes added by previous rule applications; or it can be part of a single shape added previously. Thus, shape emergence is not hierarchical. In particular, emergent shapes are not limited to ones that are "higher" or "lower"; "simple" or "complex". Shape computation can be used in design exploration to generate "emergent" designs. Such designs can range from new but conventional designs, which can be generated more readily with computation than without it, to unconventional designs that are difficult to create non-computationally (Knight, 2003a).

Overall, shape grammars should emulate the creative processes of people when they adopt and follow rules to create options or break the rules only to create and follow new ones (Knight, 2003a). In particular, humans generate options which are not stored previously in their minds. Rather, options emerge from the creative processes of humans. These creative processes can be regarded as a kind of informal computation in which designs are transformed into other designs by adding, erasing, or redrawing shapes. In each step of this process, the designer may recognize entirely new and unanticipated emergent shapes in a design, and then use them in subsequent transformations of the design. As a designer works, the design "talks back" to the designer, the designer "reframes her or his view of the design, and then proceeds accordingly. The design process is continuous, fluid and nonhierarchical (Knight, 2003a; Schön, 1987). It has been argued that it is the interaction of form that often leads to creative designs by allowing new shapes to emerge (Coyne and Newton, 1989). Important enablers of infinite spatial emergence are summarized in Table 9 below.

Table 9. Enablers of infinite spatial emergence

Characteristic	Enabler
Spatial	Shape grammars deal with subshapes such as points, lines and planes, not just higher level shapes which are fixed primitives such as circles, squares.
Infinite	Emergent shapes in any step of a shape grammar computation can be limitless because shapes are freely decomposable into subshapes.
Emergent	Shape grammars can emulate human creative behaviour because unanticipated shapes, comprising subshapes, are recognized and then used.

3.7 Development of Generative Production Systems

There are several factors which suggest that shape grammars could inform the development of generative productions systems for sustainable product creation. First, it has been argued that shape grammars differ from other types of production system formalisms in their capability for direct handling and reasoning about geometry and ability to operate on a parametric geometric representation. Also, grammars allow labels to be associated with shapes to carry non-geometric information and guide the generation process (Agarwal and Cagan, 2000). Indeed, has been argued that shape grammars provide a unifying framework for all expert systems for geometry-based engineering domains (Agarwal and Cagan, 2000).

Second, shape grammar computation generates options in a way that emulates human creative behaviour, and hence can generate infinite options for design and/or production through the emergence of shapes that cannot be predicted in advance (McCormack and Cagan, 2002; Orsborn et al., 2006). Thus, when compared to the computation of shape grammars, conventional CAD/CAM software can be considered to be passive tools, which typically document modifications that are imposed by external means such as the user's decisions. By contrast, computation of shape grammars is active in generating options that are not stored previously in a computer. These options do adhere to requirements (the grammar), but involve little, or no, input by the user (Chase, 2002). As a result, the computer becomes a generator rather than an assistant (Sheaet al., 2005). Third, shape computation can be carried out continuously throughout days and nights. Thus, shape computation can generate many options might not have been generated by an expert human using CAD/CAM software due to lack of time etc. Fourth, as summarized in section 3.3, shape grammars can enable the digitally-driven design, manufacture and assembly of a many different types of products ranging from buildings to MEMS.

Overall, it can be argued that the potential for infinite spatial emergence through the computation of shape grammars offers new opportunities for exploration and innovation in elicitation, design and production. However, it is important to note that there are alternative approaches, such as genetic algorithms, which can be used to generate form. It is also important to note that there remain unresolved challenges to the implementation of shape grammars. Accordingly, it may be more likely that shape grammars can inform, rather than solve, the development of Generative Production Systems for sustainable production creation.

Four reasons for making reference to shape grammars when developing Generative Production Systems are summarized in Table 10.

Table 10. Reasons for making reference to shape grammars.

Characteristic	**Summary**
Accessibility	More accessible to non-experts than other approaches such as genetic algorithms.
Transferability	Provides an introduction to fundamental concepts, such as spatial relations and transformation rules, which are also important to other approaches.
Versatility	Can be combined with other types of grammars in the development of robust Generative Production Systems for a wide variety of product types.
Functionality	Can be combined with other computational methods to develop Generative Production Systems with advanced functionality such as optimization.

First, grammars can be more accessible to non-experts than alternative approaches (Chase, 2005). Second, consideration of grammars can provide an introduction to important concepts, such as spatial relations and transformation rules (Chase, 2005). These fundamental concepts are important to other approaches to shape computation (Ceccato, 2009), which initially may not be as accessible a shape grammars. Third, shape grammars can be combined with other grammars, such as graph grammars, in the development of robust Generative Production Systems for a wide variety of product types (e.g. Seo, 2007; Soman et al., 2003). Fourth, shape grammars can be combined with other computational methods in the development of Generative Production Systems with a range of functionality (e.g. Grobler et al., 2008; Shea et al., 2005).

As sustainable product creation depends upon the broadest possible range of populations being able to reduce their environmental impacts, express their potential, and meet their needs; these four factors are very important. Accordingly, the formulation of entire shape grammars and their computation were the focus of the research reported here. However, it is not the purpose of this report to advocate, or suggest, that shape grammars are the only approach that should be used to develop

Generative Production Systems for sustainable product creation. Hence, the research encompassed the computation of rules, as well as the computation of entire grammars, and the combination of shape grammars with other computational methods. Further, the limitations of shape grammars, and how they might be addressed to better enable the development of Generative Production Systems for sustainable product creation, were investigated.

4. Formulation of grammars

4.1 Overview

In this section, some details are provided about the formulation of shape grammars. The details provided offer some insights into the formulation formal grammars in general. Much of the shape grammar literature is focused on the generation of designs. Accordingly, the details provided here make more reference to design than to manufacture and assembly issues.

Formulation of a shape grammar involves defining a *vocabulary* of basic forms, and defining *rules* for manipulation of those basic forms. The basic forms can be defined from a sample, for example, of a product type or of a building type. The rules define *spatial relations* between the forms and how the forms are related to each other. Shapes can exist as points, lines, planes, volumes, or any combination thereof. Shapes in the vocabulary are combined to form *initial shapes*. Shape rules can be applied recursively to initial shapes to generate options for design and/or production. This often involves generating multiple alternative options through the production of complicated forms. This is done by manipulation of the basic forms that have been defined in the shape grammar vocabulary. The manipulations are carried out in accordance with the shape grammar rules. The manipulations can be carried out by people or by computers (Stiny, 1980; 2006; Stiny and Gips, 1972).

4.2 Formulating vocabularies

Within shape grammars, a vocabulary is a limited set of shapes, no two of which are similar (Stiny, 1980). Formulation of shape grammar vocabulary can begin with analysis of existing sets of designs. In such cases, existing sets of designs provide the corpus to define vocabulary and to infer rules. The resultant shape grammar should reveal the common underlying features of the designs in the corpus. Also, it should provide the criteria to determine whether a product or building is a design within the

same design language as the corpus that has been analysed. In addition, it should specify how to generate new designs in that language (Stiny and Mitchell, 1978). Moreover, the grammar can be used to generate new design options in the same style as the existing sets of designs (Duarte, 2005).

There are a number of issues to consider in the analyses of designs and their subsequent decomposition into vocabularies of shapes. For example, anomalies in sets of designs, such as a product that is unlike all others in an industrial designer's work, should not be included in the corpus. Also, it is important to note that existing sets of designs may be closely tied to the design technologies that were used in their production. Their basic forms, for example, may have been drafted using French curves, a triangle, a T-square and so on. This raises the issue of whether a shape grammar should also be closely tied to the limited number of forms that could be produced with such kinds of old technologies (Phillips, 2008). Also, it is important to note that decomposition of designs into shape that make up a vocabulary may not be straightforward. Design drawings, for example, can contain small errors and inconsistencies. Further, constructed buildings and manufactured products can often deviate in minor ways from design drawings (Stiny and Mitchell, 1978). Some vernacular building architecture, for example, is not described within a corpus of drawings or other records. Accordingly, field study may be required to establish the corpus (Colakoglu, 2005).

Another issue is that different decompositions can could arise from analyses in different engineering domains such as electrical and mechanical (Agarwal and Cagan; 2000). Further, many products do not have obvious form-function decomposition. Using conventional design practices, this can limit human designers to only a small number of previously proven configurations. This in turn makes radical changes and rapid performance improvements difficult. In addition, a small change in specifications may require a significant time investment. Addressing this issue within shape grammar research, a coupled form-function shape grammar has been formulated using micro-electromechanical systems (MEMS) as the example (Agarwal et al., 2000).

Another important issue is the structuring of decompositions. Well-known structured decompositions are hierarchies. These are often tree-like structures with the original shape at the top and the minimum elements (i.e. atoms) at the bottom. Thus, hierarchies describe designs and their components as sums of atoms. It has been argued that while hierarchies can be useful when conceiving the materialization of an already completed design, they can be inadequate for the design process itself. This is because, it is argued, hierarchies do not support conjunction, and conjunctive combinations of entities are important in design, especially in the early stages of the process. In particular, they yield new entities that have new properties not necessarily recognized by the original entities. This supports discovery, which is an indispensible ingredient of the creative phases of

the design process. Accordingly, a topological decomposition of shape can be more useful than a hierarchical decomposition because a topological decomposition of a shape satisfies the unique representation requirement by providing for each part of the shape a pair of elements that bounds it (Krstic, 2008). On the other hand, it has been argued that the ordering of subshapes into a hierarchy can provide an efficient shape-search order during computation (McCormack and Cagan, 2002). A summary of issues to consider when formulating vocabularies of shapes is provided in Table 11.

Table 11. Issues to consider when carrying out analyses of existing sets of designs.

Issue	**Summary**
Anomalies	Designs which are unlike others in a set of designs should not included in the corpus.
Age	Existing sets of designs may be too closely tied to the limited number of forms that could be produced with old technologies.
Inaccuracies	Design drawings can contain errors and have some inconsistencies when compared to the constructed building or manufactured product.
Incompleteness	Some vernacular buildings may not be described in drawings or other types of records.
Obscurity	Many engineering products, for example, do not have an obvious form-function decomposition.
Structure	Hierarchies can be useful when conceiving the materialization of an already completed design but they can be inadequate for the design process itself.
Subjectivity	People can formulate shape vocabularies based on their own personal experiences and intentions.

It is important to note that shape vocabularies can be formulated by people subjectively defining vocabulary and rules based on their own personal experiences and intentions. This could lead to different shape grammars being formulated by different people who start from the same sample of designs. To prevent this, objective methods need to be developed for the formulation of shape grammars (Mackenzie, 1989). Accordingly, statistical decomposition has been tested within research as an alternative to intuitive or learned decomposition by people. In particular, a statistical method called principal component analysis (Shmueli et al., 2007) has been used in the formulation of shape grammar for motor vehicles (Orsborn et al., 2008) with the aim of automating the creation of shape grammar rules (Orsborn et al., 2008a). Principal component analysis produced chunks of curves which provided the basis for shape grammar vocabulary (Orsborn et al., 2008b). The term, chunks of curves, can be explained through another use of, chunks, within product design. That is: the use of the word "chunks" to refer to

major physical building blocks of a product (Ulrich and Eppinger, 1995). Each chunk is made up of a collection of components that implement the functions of the product. Similarly, chunks of curves are collections of curves that realize the style of a product. In particular, representative curves can be combined into foundational chunks. With regard to the definition of shape grammar rules, by selecting principal components with related curves, a sequence was automatically created that maintained the shape relationships found through the principal component analysis (Orsborn et al., 2008b). The chunking of curves can reduce the number of rule applications required to generate designs but, on the other hand, can reduce design flexibility.

Within research, shape grammars have been formulated to investigate relationships between similar design types; rather than just sets of designs. Similar design types can include, for example: products or buildings designed by one person over several decades; products designed over several decades by different industrial designers within the same product brand or buildings designed over several centuries by different architects within the same cultural tradition. Historical developments across similar design types can be viewed as a series of style transformations. These style transformations may have been the result of corporate planning: for example, modifications to the well established style of a branded product. On the other hand, style transformations may have been more individual and spontaneous: for example, modifications to an inherited tradition of building architecture. In such cases, shape grammars can offer a framework for mapping style transformations that have occurred over years, over decades, or even over centuries. Subsequently, modification of a shape grammar formulated during analysis of similar design types is possible through addition, deletion or modification of grammar rules. This can involve shape replacement or modification of spatial relations (Knight, 1994). In this way, it is possible to generate multiple alternative contemporary design options based on, for example, an established product brand identity or based on historical building architecture (Ahmad and Chase, 2004).

Also, design types can be analysed across product brands or across architectural traditions. Within research, for example a sample of forty-two different motor vehicles has been analysed. These different motor vehicles belong to three different classes: coupes, pickups and sports utility vehicle (SUVs). Moreover, they had been designed within several different brands. Analysis involved separating vehicle classes into views and dimensions. Analysis across brands or traditions can enable formulation of shape grammars that can be applied to generate cross-over designs and generate new categories of designs (Orsborn et al., 2006).

Shape grammars can also be original, rather than based on analyses of existing designs. Such original shape grammars are intended to generate instances of original styles of designs. On the other hand, partial shape grammars comprise a reduced set of

grammar rules with the intent to modify existing designs, rather than generate a complete design from nothing (Deak et al., 2006). A summary of different scopes for shape grammars is shown in Table 12 below.

Table 12. Different scopes for shape grammars within research.

Scope	**Examples**
Modifying existing designs	Partial adaptation of a design
Adding to sets of designs	One particular set within product type or building type
Extending existing design types	Product type within a brand or building type within a tradition
Developing cross-over designs	Different classes of product type or building type
Developing original designs	New styles for product types or building types

4.3 Defining spatial relations

As stated above, the rules in shape grammars define *spatial relations* between the forms and how the forms are related to each other. Some of the spatial relations of products can be relatively obvious. The front wheels of a motor vehicle, for example, generally sit within the front wheel wells. Similarly, the base unit of a product will typically sit below other units of the product. However, it is important to note that spatial relations can become less obvious when parts are integrated. The formulation of a shape grammar for coffee makers, for example, involved definition of three main parts: the filter unit, the water storage unit, and the base unit. These three units are arranged around the space for the coffee pot, which acts as the initial shape for the grammar. The grammar generates a complete coffee maker by first designing the base and the filter units, then integrating them together using the water storage unit.

It has been argued that different types of spatial relations between two shapes can be defined as: contain, overlap, share boundary and disjoint. A shape *a* can be said to *contain* a shape *b* if *b* is part of *a*. Two shapes *overlap* if they have a common part and neither shape fully contains the other. Two shape *share a boundary* if they do not overlap, but their boundaries do overlap. If two shapes do not contain, overlap or share boundary, they can be said to be *disjoint* (Stouffs and Krishnamurti, 1994). It has also been argued that different types of spatial relations between shapes can be defined as: *within*, *intersection*, *boundary relations*, and *continuous* (Chase, 1997). A summary of spatial relations is shown in Table 13 below.

Table 13. Types of spatial relations.

Type	Example
Spatial relations between two shapes	Contain
	Overlap
	Share boundary
	Disjoint
Spatial relations within one geometric form	Parallel
	Perpendicular
	Intersect

Also, many spatial relations can exist within just one geometric form. Consider, for example, a square. The four maximal lines in a square encapsulate a variety of spatial relationships: each line is the same length, each line is parallel to one other perpendicular to two others and each maximal line contains two intersections. Thus, even one square contains a rich variety of spatial relationships between the constituent elements — the maximal lines themselves and the intersections (Tapia, 1999). Other quadrilaterals, such as rhombus, trapezoid and rectangle, have an equal number of line segments to a square but have spatial relations which make them all distinct. A rhombus, for example, is like a square in that it has reflection symmetry about imaginary lines of symmetry through its diagonals. However, a rhombus is unlike a square in that its lines do not intersect at right angles to each other.

Spatial relations can be used to formulate a default hierarchy of shape decomposition. The levels of the hierarchy can defined so that the most constrained lines of a shape are those lines that the designer intended exactly. These most constrained lines have specified parametric relations to other line segments and those relations, if altered, will compromise the designer's intentions. Conversely, the lowest level of the hierarchy, which contains the least constrained line segments, implies only a specific connectivity between line segments. The formulation of this type of default hierarchy involves separating the lines in the shape that the designer specified exactly from lines in the shape that were intended as a general scheme. Different rules can be allocated to different levels of the hierarchy. In particular, different rules for the lines in the shape that the designer specified exactly and other rules for the lines that were intended as a general scheme (McCormack and Cagan, 2002).

4.4 Developing grammar rules

As stated above, emergence in shape grammars emulates human creative behaviour in design (Knight, 2003a). In particular, human designers generate design options which are not stored previously in their minds. Rather, designs emerge from the creative processes of human designers. These creative processes can be regarded as a kind of informal computation in which designs are transformed into other designs by adding, erasing, or redrawing shapes. Accordingly, shape grammar rules should be congruent with designers' visual intuitions (Stiny, 1980).

A shape grammar minimally consists of three shape rules: a start rule, at least one transformation rule, and a termination rule. Rules within shape grammars take the form $a \rightarrow b$, where a and b are both shapes. In order to apply this rule to shape, c, an instances of a must be located in c. If a transformation of a, $t(a)$ is located in shape c, the rule can be applied to produce a new shape c' by using the equation: $c' = c - t(a) + t(b)$. Thus, executing the action part of shape grammar rules involves subtraction and addition (McCormack and Cagan, 2006). In simple terms, rules apply in a two-stage process: (1) find a part of c that looks like a; and (2) replace this part with a new part that looks like b. The two-stages are always linked (Stiny, 1999).

Transformational rules within shape grammars are rules for spatial transformations. The types of spatial transformations include translation, reflection, rotation, scale, and shearing. Spatial translation involves moving every point of a shape a constant distance in a specified direction. Spatial reflection (i.e. mirror image) involves a transformation in which the direction of one axis of a shape is reversed. Spatial rotation involves movement of a shape in a circular motion. A two-dimensional shape rotates around a center (or point) of rotation. A three-dimensional shape rotates around an axis. Translation, reflection and rotation can be thought of a being rigid motions. In particular, no amount of translation, reflection or rotation will destroy a specific feature of a shape but only more features from place to place. By contrast, anisotropic scaling (different scaling factor in all directions) can destroy the symmetry of a shape unless the scaling is along perpendicular to the line of symmetry. Isotropic scaling (same scaling factor in all directions does not, however, affect the symmetry of a shape. Shearing leaves fixed all points on one axis and other points are shifted parallel to the axis by a distance proportional to their perpendicular distance from the axis. A summary of the types of spatial transformations enabled by shape grammar rules is provided in Table 14 below.

Table 14. Different types of spatial transformations enabled by shape grammar rules.

Type	Summary
Spatial translation	Moving every point of a shape a constant distance in a specified direction.
Reflection	The direction of one axis is reversed and so produces the mirror image.
Rotation	A two-dimensional shape rotates around a centre or point of rotation. A three-dimension shape rotates around an axis.
Anisotropic scaling	Different scaling factor is applied in all directions, and destroys symmetry unless perpendicular along line of symmetry.
Isotropic scaling	Same scaling factor is applied in all directions, and does not affect the symmetry of the shape.
Shearing	Leaves fixed all points on one axis and other points are shifted parallel to axis by a distance proportional to their perpendicular distance from axis.

Rules can make ideas explicit so they can be analyzed, changed or communicated more easily (Knight, 1999). Rules can be course or fine depending upon the level of detail within a rule. By dividing the design process into two stages, shape searches can be performed more quickly because the coarse representation contains fewer geometric entities than the fine representation. Overall, the level of detail represented in the grammar being a function of what is intended to be accomplished with the grammar (McCormack and Cagan, 2002). If performance metrics are associated with rules, it is critical that the metric depend only on the information provided by the shape rules (Agarwal et al., 1999). It is important to note that rules may be needed which do not transform shapes (Stiny, 1996). Rather, they enable subsequent transformation through the application of other rules or enable the passing of geometric information to simulation / analysis software. Such rules may be expressed as $a \rightarrow a$, rather than $a \rightarrow b$ (Agarwal and Cagan, 2000).

The formulation of rules within research has led to a number of useful insights. First, it has been argued that creativity in rule-based design lies in the formulation of the rules (Colakoglu, 2005). Further, it has been argued that the effective organization of rules is critical for their retrieval and adaptation in rule based systems (Seebohm and Wallace, 1998). This can be facilitated by preparation of a rule dependency graph to elucidate relationships between shape grammar rules. In particular, reference to a rule dependency graph can clarify how changes to one rule may affect other rules. For example, it is important to understand which characteristics of a design appear in more than one view and how to link them together through their rule applications (Orsborn. et al., 2006). Generally, a modular organization of rules can facilitate the addition, deletion and/or replacement of sets of rules. Devising a visual representation of product /

building components can be important to facilitate the formulation of rules. Further, understanding the intrinsic orders / logical progressions of processes can be important. Accordingly, the formulation of rules can be informed by reference to, for example, handbooks that describe processes for design and/or production. Reference to such texts can provide useful guidance into the structuring of rules into groups which are congruent with effective processes. Groups of rules can, for example, cover process stages such as: locate functional zones; define circulation scheme; divide zones into rooms; introduce details; introduce openings (Duarte, 2005). Stages can correspond more or less to natural and intuitive design processes (Stiny and Mitchell, 1978).

However, it is important to note that rules can be applied in novel orders during shape grammar computation. This can be done to explore novel alternatives. Rules can be constrained so that designs are guaranteed to be within a feasible design space, no matter how the rules are applied. Thus, certain assumptions and constraints have to be imposed on the rule selection algorithm to constrain the design space. These assumptions and constraints need to be carefully determined so they do not excessively narrow the design space, but do prevent searching a space of infeasible designs. Overall, formulation can be a gradual process with numerous problem solving sessions and example problems (Soman et al., 2003).

A parallel grammar is a network of two or more grammars that operate in conjunction with each other. One grammar for building architecture could generate floor plans; another could generate elevations; another could generate sections; another could generate numerical descriptions. The rules of parallel grammar may be linked so that the application of a rule in one grammar triggers the application of one or more rules in other grammars. The generation of first floor plans, for example, could drive the generation of upper floors. Then, the elevations which are generated would be determined by the layout of floors (Duarte, 2005). Similarly, a floor plan grammar could be linked to a site plan grammar. When the rules of a parallel grammar are linked, the linked rules can be expressed as one compound rule. The component rules of a compound rule may or may not operate in the same space or domain (Knight, 2003a). Parallel grammars are defined in terms of composite algebras (Chase and Ahmad, 2005; Knight, 2003b).

It has been argued that it may be possible to: identify priorities of rules; calculate preference values for each rule based on frequency of use; and use preference values as speculative tools to provide customizable categorizations of design outcomes. In this way, the outcomes from applications of rules could be limited in accordance with the particular preferences of particular users (Lim et al., 2008). It can be possible for a shape grammar system to enable its users to carry out both rule development and rule application (Tapia, 1999). It is important to note that rules for a shape grammar which enable physical production need more information and more illustration during

formulation (Sass, 2007b). A summary of considerations in the formulation of shape grammar rules is shown in Table 15 below.

Table 15. Considerations in the formulation of shape grammar rules.

Consideration	**Relevant technique**
Organization for retrieval etc	Rule dependency graph
Intrinsic orders of processes	Process guides
Assumption and constraints	Problem solving with examples
Duration of computation	Course rules supported by fine rules
Level of emergence	Optional rules without explicit dimensions
Dependencies among components	Parallel grammars defined in composite algebras
Particular user preferences	Preference values as speculative tools
Enable physical production	More information and illustration during formation

The three kinds of emergence in a shape grammar can be determined during the definition of rules: anticipated, possible and unanticipated. With anticipated emergence, the authors of a grammar know that certain shapes will emerge from their application. In anticipation of these emergent shapes, rules that operate on them are included in the grammar. Anticipated emergence is key to analysis applications of shape grammars. In analysis, emergence is necessary but must be anticipated so that only a carefully circumscribed range of new designs is generated. Rules must generate "all and only" designs of a specified kind (Knight, 2003a). Such grammars can be thought of as set grammars (e.g. Heisserman and Woodbury, 1994). Nonetheless, designs in computation may have global characteristics that cannot be predicted from the local actions of rules. With possible emergence, the authors of a grammar write rules and think that, perhaps, certain shapes might emerge. Rules are included that apply to these possible shapes just in case they do emerge. With unanticipated emergence, the authors of grammar write rules and compute with them. Shapes emerge that were not premeditated or anticipated in any way. In order to compute with these shapes, the grammar may need to be updated with new rules (Knight, 2003a). Unanticipated emergence is pervasive in the processes of human designers, particularly in the early, conceptual stages of design when the open exploration of novel, unexpected design ideas is important. Unanticipated emergence makes much of design happen (Knight, 2003a). A shape grammar can be said to be informal when its rules are generally optional; and when relative dimensions are not made explicit in the rules (Phillips, 2008). A summary of types of emergence arising shape grammar rules is provided in Table 16 below.

Table 16. Types of emergence arising from shape grammar rules.

Emergence	Rules
Anticipated	Rules generate all and only designs of a specified kind (i.e.set grammars).
Possible	Rules are included that apply to possible shape which might emerge.
Unanticipated	Rules may be optional and/or may need to be updated.

4.5 Define initial shapes

As stated above, shapes can be defined and manipulated within a Cartesian coordinate system. That being a system for locating and measuring points in space based on a three dimensional axis labeled X, Y, Z. Thus, the initial shape can be, for example, the origin of the Cartesian coordinate system (Stiny and Mitchell, 1978). Also, the initial shape can be a relative coordinate system which, for example, links the front, rear and side views of a car (Orsborn, Cagan, Pawlicki, and Smith, 2006). Coordinate system can have real axes (i.e. every point on the axes corresponds to a real number) and an associated euclidean metric (Stiny, 1980). It is beneficial, in some cases, for the initial shape to be a shape which is congruent with the typical starting place for physical production, such as a foundation (Sass, 2007b). The formulation of a shape grammar for coffee makers, for example, involved definition of three main parts: the filter unit, the water storage unit, and the base unit. These three units are arranged around the space for the coffee pot, which acts as the initial shape for the grammar. The grammar generates a complete coffee maker by first designing the base and the filter units, then integrating them together using the water storage unit.

4.6 Language of shape

A shape grammar is defined over a set of shapes, and maps shapes into shapes to generate a language of shapes (Stiny, 1976). Language of shape is considered in the shape grammar literature in the context of design. For example, it is proposed that shape grammar rules should be explicit representations of the underlying properties and structures of sets of designs, thus shifting emphasis away from individual designs to languages of design (Stiny, 1980). A language of designs has been described as a set of descriptions of individual designs: with such descriptions enumerating the component elements one-by-one in a finite sequence of symbols. For example, spatial designs – architectural, mechanical, or electronic ones - are often given by drawings which consist of lines. These drawings can each be encoded by a finite sequence of symbols by listing the end points of the lines in them that are not parts of longer lines. In general,

descriptions of designs are often given by diagrams, drawings, or schematics. Such descriptions can always be given by a finite set of symbols (Stiny and March, 1981). Languages of design can be defined in accordance with the steps outlined in the proceeding sub-sections: formulate vocabulary of shapes; determine spatial relations between shapes in the vocabulary; develop rules; define initial shape. Overall, each shape grammar defines a language of designs (Stiny, 1980). Languages of designs may be thought of as corresponding to certain "styles" of design (Stiny and March, 1981).

It has been argued that when designers use a language of designs, they can avoid the combinatorial difficulties of ad hoc design development which lead to the consequences of succeeding changes becoming more and more obscure. Also, it has been argued that rules can be modified systematically to define new languages of design which reflect changing circumstances or incorporate new ideas (Stiny, 1980).

More broadly, formal languages are important within the fields of logic, computer science and linguistics. An important practical application is for the precise definition of syntactically correct programs for a programming language. Apropos, shape grammars need to be precisely defined to enable their computation.

5. Computation of grammars

5.1 Overview

In this section, some details are provided about the computation of shape grammars. These details offer some insights into the computation of formal grammars in general. First, the definition of shape grammars with algebras is outlined. Then, enabling algorithms are discussed. These include standard geometric algorithms as well as those which have been formulated specifically to enable computation of shape rules. Next, a summary of their description with pseudo-code is provided. Subsequently, various aspects of shape grammar implementations with software are described. In conclusion, examples are given of computational methods which have been combined with shape grammars, such as ontology and optimization techniques. Much of the shape grammar literature is focused on the computation of designs. Accordingly, the details provided here make more reference to design than to manufacture and assembly issues.

5.2 Definition with shape algebras

As shown in Table 17, within shape algebras, letters or other symbols are used to represent shapes – rather than numbers (Stiny, 1991). Such algebras provide the framework for computation of shape grammars (Knight, 2003b). For example, the definition and application of rules is enabled by shape algebras (e.g. Krstic, 2001) with subshapes being defined with algebras so their geometries can be constructed, recognized, and/or reconstructed through computation (Krishnamurti, 1980; Krishnamurti and Earl, 1992).

Table 17. Foundations of shape algebra.

Algebra	**Objects**	**Basic Elements**		**Boundaries**
	Shapes determined by finite sets of maximal	Defined in dimension	Have finite non-zero	Every basic element has a boundary belonging to
U_0	Points	0	-	-
U_1	Lines	1	Length	U_0
U_2	Planes	2	Area	U_1
U_3	Volumes	3	Volume	U_2

As shown in Table 18 below, algebra U_{ij} be presented as a hierarchy (e.g. Knight, 2003b; Krstic, 2001); with elements of dimension $_i$ in a space of dimension $_j$. Thus, U_{01} describes a point on a line; and U_{23} describes a plane in a volume.

Table 18. Shape algebra U_{ij}.

Dimension $_i$	**In a space of dimension $_j$**			
	Point	Line	Plane	Volume
Point	U_{00}	U_{01}	U_{02}	U_{03}
Line		U_{11}	U_{12}	U_{13}
Plane			U_{22}	U_{23}
Volume				U_{33}

The algebras in the tables are ordered recursively via basic elements. The ordering highlights their underlying properties in terms of the embedding relation, finite content (non-zero length, area, and volume) for lines, planes, and solids, and definite boundaries for these elements that are shape but not parts. This makes computer implementations of shapes possible, ultimately with points that have neither content nor boundaries. Moreover, it lets computers calculate with shapes as if this were done by hand and eye (Stiny, 1999). By contrast, shapes in traditional computer graphics systems are often represented as sets of lines that cannot be further decomposed, i.e. no portions of lines can be easily recognized. Shape algebraic representations act against such restrictions, as they require only minimal predetermination of structure (Chase, 1996).

If two algebras are combined in a Cartesian product, then a reciprocal relationship can be formed (Stiny, 1991) in which rules defined in one algebra (e.g. U_0) control the use of rules defined in another algebra (e.g. U_1). Generally, Cartesian product can be thought of as the set of elements common to two or more sets; "the set of white cars is the intersection of the set of cars and the set of white things". A set is a group of things

of the same kind that belong together and are so used; "a set of books"; "a set of golf clubs"; "a set of teeth". More precisely, Cartesian product can be described as the collection of all ordered pairs of two given sets such that the first elements of the pairs are chosen from one set and the second elements from the other set: this procedure generalizes to an infinite number of sets. Within shape computation, a point can be chosen from U_0 shape vocabulary, for example, can be associated with a line chosen from U_1 shape vocabulary. The point can control the use of the line, for example, by determining what rotations of the line can be executed. With regard to the formulae, $a \rightarrow b$, and, $c' := [c - t(a)] + t(b)$, if the use of rules are to be controlled by augmenting a in a Cartesian product, then the devices used for this purpose must be incorporated in c. This requires that the object that begins the computation and right hand side of the rules (i.e. b) also be augmented. How this is achieved varies from one case to the next (Stiny, 1991).

Every algebra of shapes, $U_{ij,}\ 0 \leq_i \leq_j$, has most of the properties of a Boolean algebra (Stiny, 1993). The term, Boolean algebra, honors George Boole (1815–1864), a self-educated English mathematician. In abstract algebra, a Boolean algebra, or Boolean lattice, is a complemented distributive lattice. In mathematics, a lattice is a partially ordered set in which any two elements have a unique supremum (the elements' least upper bound; called their join) and an infimum (greatest lower bound; called their meet). Lattices can also be characterized as algebraic structures satisfying certain axiomatic identities. Since the two definitions are equivalent, lattice theory draws on both order theory and universal algebra. Moreover, this type of algebraic structure captures essential properties of both set operations and logic operations. An operation being an action or procedure which produces a new value from one or more input values. In particular, every algebra of shapes, $U_{ij,}\ 0 \leq_i \leq_j$, is a relatively complemented distributive lattice (Stiny, 1993). In the mathematical discipline of order theory, a complemented lattice is a bounded lattice in which every element a has a complement, i.e. an element b satisfying $a \vee b = 1$ and $a \circ b = 0$. A relatively complemented lattice is a lattice such that every interval is complemented. Complements need not be unique.

An algebra U_{ij} can be augmented by associating labels from a given vocabulary with basic elements to classify them in shapes, or to introduce additional information. This allows for shapes to be formed as collections of others that are distinguished by distinct labels. Members of different collections interact if they are distinguished by the same label, and are independent otherwise. A new algebra V_{ij} is so defined that preserves the Boolean properties of U_{ij} , too, if labels are invariant under the transformations (Stiny, 1992). Labels may be given simply to identify and classify points, or they may have a semantics allowing them to carry important information about shapes and other things (Stiny, 1991). Furthermore, weights can be introduced in an algebra U_{ij} to obtain a new algebra W_{ij} (Stiny, 1992). Weights are familiar in traditional art and design, where lines

of different thicknesses (i.e. weights) are used in drawing. However, weights in shape computation can encompass a much broader range of properties. Properties can be aesthetic, functional, structural, and so on. For example, points can have diameters; lines can have thicknesses; planes can have colours; solids can have materials; and so on. In a study involving generation of micro-electromechanical systems (MEMS), the four main elements (central mass, actuators, anchors, and springs) were each assigned a different weight to ensure that feasible designs were generated (Agarwal et al., 2000). A summary of the different algebras is provided in Table 19 below.

Table 19. Applications for different algebras.

Algebra	**Application**
U_{ij}	Shapes determined by finite sets of maximal points, lines, planes, volumes
V_{ij}	Labels to identify and classify points, or carry important information
W_{ij}	Weights to include properties, e.g. aesthetic, functional, structural, in shapes

The Cartesian product of algebras can be represented in a matrix. In particular, parallel grammars defined in terms of composite algebras (Knight, 2003b) are suited to this representation (e.g. Duarte, 2005). As stated above, a parallel grammar is a network of two or more grammars that operate in conjunction with each other.

5.3 Enabling with algorithms

As stated above, shape grammars are examples of *production system formalisms* (Post, 1943). These are computer programs consisting of a knowledge base of rules and general facts, a working memory of facts concerning the current case, and a rule interpreter within an inference engine. Rule interpreters generally execute a forward chaining algorithm for sequences of rule applications. Forward chaining starts with the available data in the working memory and searches the rules until it finds one where the condition (e.g. IF statement on LHS) is known to be true. When found it can infer the action (e.g. THEN statement on RHS), resulting in the addition of new information to its working memory. Then, searching the rules continues.

An algorithm is a finite sequence of instructions that provide an explicit, step-by-step procedure for solving a problem. It is formally a type of effective method in which a list of well-defined instructions for completing a task will, when given an initial state, proceed through a well-defined series of successive states, eventually terminating in an end-state. An effective method for a class of problems being a method for which each step in the method may be described as a mechanical operation and which, if followed rigorously, and as far as may be necessary, is bound to work for all instances of

problems of the class. Algorithms have been formulated to enable shape rule applications (Krishnamurti, 1980; Krishnamurti and Earl, 1992; Krishnamurti and Stouffs, 2004). In addition, standard geometric algorithms can be used to enable shape computation. De Casteljau's algorithm, for example, has been used in curve-matching during shape recognition (McCormack and Cagan, 2006). De Casteljau's algorithm is a recursive method to evaluate polynomials in Bernstein form or Bézier curves. The de Casteljau's algorithm can also be used to split a single Bézier curve into two Bézier curves at an arbitrary parameter value. Also, Lagrange's theorem has been suggested to classify the possible actions of $a \rightarrow b$ in terms of the symmetries of the shapes a and b (Stiny, 1999). Lagrange's Theorem simply states that the number of elements in any subgroup of a finite group must divide evenly into the number of elements in the group. A consequence of Lagrange's Theorem would be, for example, that a group with 45 elements could not have a subgroup of 8 elements since 8 does not divide 45. It could have subgroups with 3, 5, 9, or 15 elements since these numbers are all divisors of 45.

It is important to appreciate that carefully formulated data structures are needed for efficient operation of algorithms. When De Casteljau's algorithm has been used in curve-matching during shape recognition, for example, shapes have been decomposed into a hierarchy of subshapes ordered by decreasing restrictions on their spatial relations. A high level of restriction on spatial relations could be, for example, that lines intersect perpendicularly and are the same length. A lower level of restriction could be, for example, that lines intersect. This type of ordering enables subshapes of a whole shape to be dealt with individually when performing the subshape recognition process, and yet can allow an entire shape to be parametrically recognized through their combination. Thus, the ordering of shapes into a hierarchy based on spatial relations can provide an efficient shape-search order (McCormack and Cagan, 2002). Further efficiency in the parametric recognition of curved-line shapes can be achieved by employing a two-step approach. The set of potential matches can be found first on the basis of a representative straight-line shape (i.e. coarse matching). Then, potential matches can be validated or rejected by comparing actual curve (fine checking). This approach has advantages over the pure matching of characteristic polygons in that it can match equivalent curves with differing characteristic polygons, in addition to emergent shapes. The representative straight-line shape can be a collection of lines connecting distinct points from the curvy shape. The set of distinct points includes intersections between curves, curve endpoints, and projected intersections of well-know curves such as circles. Endpoints can be used as distinct points for matching otherwise indeterminate shapes. Fine checking can be performed using a number of different methods. Control polygons can be compared for equivalence if the number of control points defining each curve is the same. Control points can be added to a polygon with fewer points without changing the curve, in order to allow for comparison (McCormack and Cagan, 2006). It

is within such carefully formulated data structures that algorithms can operate efficiently. A summary of important issues relating to algorithms is provided in Table 20 below.

Table 20. Issues in the formulation of algorithms.

Issue	Approach
Chaining	Forward chaining
Sources	Shape grammar literature; standard geometric algorithms
Data structures	Decompositions and rules need to facilitate efficient operation of algorithms

5.4 Description with pseudo-code

In order to facilitate the planning of computer program development, algorithms can be expressed as pseudo code before actual coding is carried out. Typically, pseudo-code will provide a description of the key principles of an algorithm using words of natural language such as IF and THEN together with some compact mathematical notation. Pseudo-code descriptions are not intended to be machine readable, and can be much easier for a broad range of people to understand than programming language code. Also, pseudo-code descriptions can be read more quickly than programming language code because details are omitted which are not essential for human understanding of the algorithm, such as variable declarations, system-specific code and subroutines.

Although pseudo-code will seldom obey the syntax rules of any particular programming language; style and syntax may be borrowed from some conventional programming language such as BASIC, C++, Java, and Lisp. Thus, the use of pseudo-code can avoid many of the ambiguities common in natural language statements, while remaining independent of a particular implementation language. Examples of pseudo-code for algorithms formulated to enable shape rule applications are presented in the literature (e.g. McCormack and Cagan, 2006; Soman et al., 2003).

5.5 Implementation with software

Programming languages enable expression of algorithms in a form that can be executed by a computer. Shape grammars have been implemented in a variety of languages including: ACIS, AutoLISP, C/CLP, CLIPS, C++, Java, LISP, Perl, PROLOG, and SAIL (Chau et al., 2004). One interpreter, for example, has been implemented in C++; and was compiled and run on a Windows workstation (McCormack and Cagan, 2006).

The need for effective user interfaces for the generation, visualization and manipulation of shapes has led to the use of 3D graphics APIs such as Open Inventor in conjunction with, for example, Java (Wang and Duarte, 2002). The potential of scripting languages, such as ActionScript, to improve user-interactivity has been recognized (Wu, 2005). Scripting languages enable the encoding of new functionality into an existing program, rather than developing new software from scratch (Loukissas, 2003). Thus, more time can be spent focusing on function, structure, and interface, rather than focusing on coding and testing (Wu, 2005). There are limitations to what can be scripted in any given software environment. However, scripting provides sufficient access to underlying structures to enable development of personal and/or project-specific tools. Such tools can be developed so they are more easily usable by a wide range of people who may be involved in carrying out a relatively narrow range of tasks. It has been argued that scripting languages can offer massive increases in productivity with little or no negative effects on eventual system performance (Greiner, 2008). RhinoScript, for example, is the scripting language of the 3D CAD software Rhineceros (Rhino), and has been used in explorations of shape computation involving implementation of a few rules (Loukissas, 2003).

Shape grammars can be implemented as an extension to design drafting software, such as AutoCAD (Romeo, 2002, cited in Loukissas, 2003), and integrated with parametric design software, such as Generative Components (Shea et al., 2005). Drafting software can be thought of as "traditional" CAD systems because they can be seen as evolved, automated versions of the traditional drafting table of the architect or engineer. By contrast, parametric design software allows users to define geometrical entities by establishing relationships of geometrical or mathematical dependency between different elements of the design (Cardoso and Sass, 2008). In either case, there are challenges in shape grammar implementation with CAD software. For example, a major challenge to shape grammar implementation with parametric CAD is that this type of software involves top-down manipulation in which changes in the general shape are propagated to the parts. By contrast, the computation of shape grammars involves bottom-up changes on the overall shape by the addition and subtraction of parts. Nonetheless, a generative structural design system called eifForm has been combined with the parametric CAD software package called Generative Components. This has been achieved by use of "federated" systems architecture and the employment of eXtensible Markup Language (XML) for integration. XML can provide flexible and adaptable information identification because it is a meta-language, i.e. a language for describing other languages that allows individuals to design their own markup languages for almost any type of document. XML is a human-readable open standard with off-the-self parsers for common programming languages such as C, C+, C++, Java, Perl and Python. Flexibility and interoperability make XML a useful format for 2D

web-graphics, 3D modelling and VRML (virtual reality mark-up language) rendering (Shea et al., 2005). The motivations for integrating eifForm with Generative Components include: development of richer input models; more effective visualization and manipulation of geometric and topological relations; improved interplay between generated shapes and whole design (Shea et al., 2005). A summary of issues relating to implementation with software is provided in Table 21 below.

Table 21. Issues in the formulation of algorithms.

Issue	**Approach**
Programming	ACIS, AutoLISP, C/CLP, CLIPS, C++, Java, LISP, Perl, PROLOG, SAIL, Open Inventor, ActionScript, Rihnoscript have been used.
CAD	Use of "federated" systems architecture and the employment of eXtensible Markup Language (XML) for integration.
CAD/CAM	Innovations, such as CAD scripting, file formats like STL, and bilateral contouring, can facilitate streamlining of design and production.
Web	Innovations, such as Extensible 3D (X3D) Graphics and gesture-responsive scripting may facilitate running of shape grammars on the internet.

Potential for combining shape grammars with CAD/CAM (computer-aided manufacturing) has been explored within research. This has led to the observation that there is a lack of software to simultaneously support design growth and design flexibility. The term, design growth, refers to the potential for designs to evolve during the application of shape grammars. The term, design flexibility, refers to the potential for design to be realized at different physical scales with different materials and different machines (Cardoso and Sass, 2008). Nonetheless, the introduction of CAD software and file formats for digitally-driven manufacturing opens up more possibilities for streamlining creative design with physical production. STL, for example, is a file format native to the stereolithography CAD software created by 3D Systems. STL supported by many other software packages; it is widely used for rapid prototyping and computer-aided manufacturing. An STL file describes a raw unstructured triangulated surface using a three-dimensional Cartesian coordinate system. Further, research has demonstrated that manufacture at two different scales with two different types of machines from the same geometric data is possible. In particular, geometric descriptions are needed that co-ordinate geometries between device types. This involves the rethinking of tool path goals and artefact assembly methods through approaches such as bilateral contouring (Sass, 2007a) and rationalizing materials so they are of one type irrespective of machine (e.g. flat solid material).

Also, the potential to streamline very creative design with physical production with, so called, CAD scripting, using computer-aided design packages such as Rhino has been recognized for some years (Sass et al., 2005). It has been argued that scripting can help relate creative design to manufacturing and assembly. This requires the tailoring of scripts for specific materials properties and machine characteristics. In order to generate a robust model for a 3D printer, for example, a script must specify only solid objects. Further, objects must be above a minimum thickness and beneath a maximum size: with the allowable thickness of a model being partly a function of the overall proportion of the model (Loukissas, 2003).

It has been proposed (Wang and Duarte, 2002) that shape grammars could run directly on the internet from web browsers that have 3D interactive user interfaces with standard file formats such as VRML (Virtual Reality Markup Language – now superseded by X3D). Also, it has been proposed that direct "gestural inputs" from users, such as scanned images, digitally drawn shapes, or 3D digitized three dimensional objects, can be enabled through "gesture-responsive" scripts. Those being scripts with a rule set open to definition in relationship to an existing shape (Loukissas, 2003).

An overview of different representations of shape grammars throughout their formulation and computation is shown in Figure 6. This shows that the representations are primarily spatial in the corpus, during formulation of vocabulary, and in the development of rules. The development of rules, the composition of algorithms and the writing of pseudo code can involve shape algebra and natural language. Implementation with software will involve the use of computer programming languages. Subsequently, the outputs of shape grammar computation will be spatial representations.

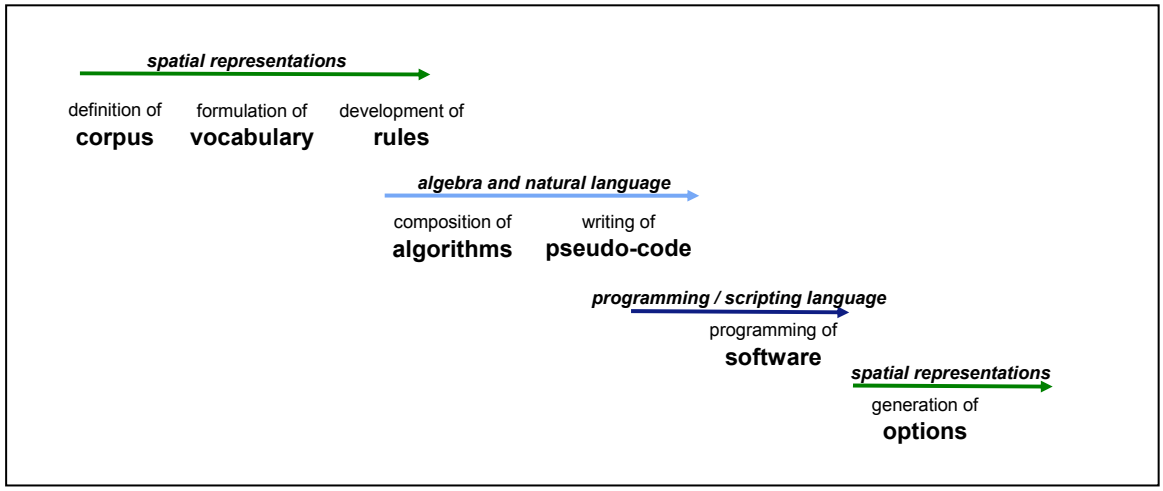


Figure 6. Representational methods.

5.6 Complementary computational methods

It has been recognized that better generative systems can be achieved if shape grammars are implemented in conjunction with other computational methods rather than implemented in isolation (Bruton, 1997). Within research, for example, shape grammar has been used in conjunction with ontology. In one study, design knowledge was captured in the ontology and processed by the shape grammar. The interaction between shape grammar and ontology included a list of questions that the shape grammar posed for a specific design type. The research suggests that ontology and shape grammar may have complementary properties that when combined can strengthen the overall process of design (Grobler et al., 2008).

Also, combining shape grammars with graph grammars can be beneficial. In simple terms, graphs are diagrams displaying data, in particular diagrams showing the relationship between two or more variables. Graph grammars consist of production rules to create valid configurations of graphs for a specific domain (Plump, 1999). In particular, the rules in graph grammars enabling match and replace operations for nodes and edges in a network. Compared to shape grammars, graph grammars offer the advantage of expressing connectivity between elements. This allows representation of non-serial component arrangements and functional relationships. Hence, graph grammars have been used in a similar way to shape grammars to design graphs for the mechanical configurations of gear boxes, electric motors and a variety of machines (Schmidt and Cagan, 1997). One study combined graph grammars and shape grammars in apartment house design (Seo, 2007). This was considered necessary because relational meanings between consisting parts are a relatively minor concern in shape grammars. By contrast, design often starts from the conception of approximate forms out of unrefined candidate configurations in human design processes. Thus, topology and form interplay from the outset to achieve, for example, building functionality in human design. In another study, a grammar was developed which operates on the nodes and arcs of a graph but generates a complete shape. The positions of the nodes within the graph directly represent elements of sheet metal (Soman et al., 2003). More generally, it has been argued that shape grammars could be combined with graph grammars to overcome the problem of shape grammars not being able to deal with CAD primitives directly (Deak et al., 2006).

Many of the outputs of Generative Production Systems will be of limited usefulness unless the generative process is directed towards specific goals. Accordingly, shape grammars have been combined with a variety of computational techniques for optimally directed search. A search conducted in this way will find near-optimal solutions without necessarily finding the optimal solution (Cagan and Agogino, 1991a; 1991b). Thus, techniques do not guarantee that exact mathematically optimal solutions are generated. Rather, optimization techniques direct the generation of shapes towards specified

objectives or criteria. These can be for design and for production. Within design, for example, elegance based on an aesthetic value determined from spatial uniformity can be important (Shea and Cagan, 1997). Within production, for example, process times and energy consumptions can be important. Accordingly, optimization could direct generation towards minimizing both time and energy (Soman et al., 2003). Optimization techniques can enable shape rules to be tested for possible application before they are actually applied. If the shape produced by a rule advances towards the specified objectives, then the rule is applied. In order to determine if a shape advances toward the objectives, the shape is evaluated and described using an external algorithm. Evolutionary algorithms, for example, have been combined with shape grammars by several research scientists. Such algorithms can be applied to automate design search and evaluation. It has been argued that evolutionary algorithms have potential for replacing the manual effort of rule selection and design evaluation associated with shape grammars. Further, it has been argued that application of such algorithms can lead to evolution of new generations of shape grammars that can produce better designs than their initial shape grammar formulated by human experts. For example, through mutation and cross-over of rules (Ang et al, 2006; Caldas, 2008; Chouchoulas, 2003; Gero et al., 1994; Lee and Tang, 2004). An illustrative overview of possible combinations of other computational methods with shape grammars is shown in Figure 7.

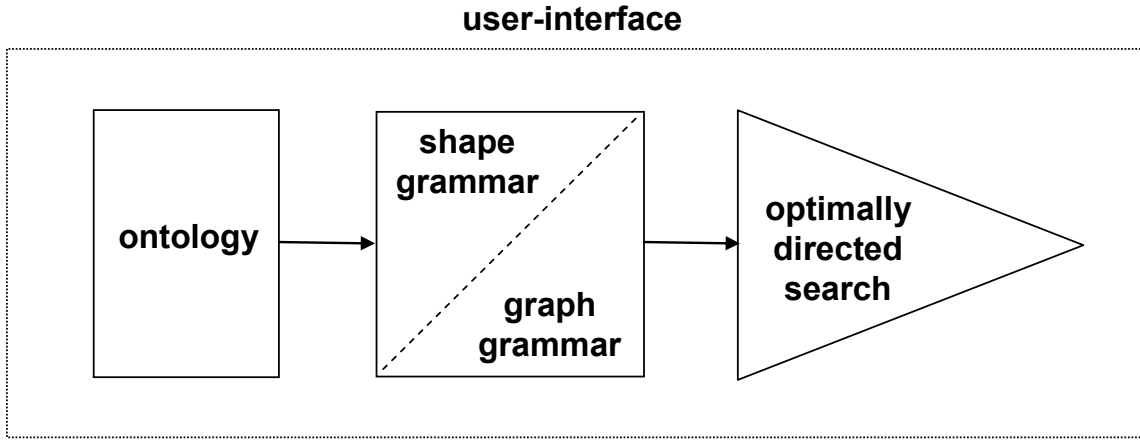


Figure 7. Illustrative combinations with other computational methods.

Shape grammars have also been combined with another computation method for optimizing search and evaluation: simulated annealing. The process arising from this combination has been called, shape annealing (Shea and Cagan, 1997; 1999). Simulated Annealing is a computational technique for finding a good solution to an optimization problem by trying random variations of the current solution. It exploits an analogy between the way in which a material is heated and then cooled for making the material

less brittle. Simulated annealing can be said to expose a "solution" to "heating" and "cooling" to produce a more optimal solution (Kirkpatrick et al., 1983). More broadly, shape optimization can be viewed as a part of the branch of computational mechanics called structural optimization. In structural optimization problems, data of the mathematical model are set up that describe the behaviour of a structure in order to find a situation in which the structure exhibits a priori given properties. In shape optimization, the optimization of the geometry is of primary interest. Shape optimization activities can be grouped into three categories. First, sizing optimization involves optimizing the typical size of a structure: for example, a thickness distribution of a beam or a plate). Shape optimization itself involves optimizing the shape of a structure without changing the topology. Then, topology optimization involves optimizing the topology, as well as the shape, for example by creating holes (Haslinger and Mäkinen, 2003). Shape annealing has been put forward as a stochastic method that combines discrete topology changes with continuous shape and sizing of topologies to generate, for example, optimally directed planar trusses (Reddy and Cagan, 1995). In particular, the shape annealing method builds structures by using a shape grammar; optimizes the structures with the stochastic optimization method of simulated annealing; and analyses the structures by using the finite element method (Shea and Cagan, 1997). Shape annealing has been shown to be capable of generating traditional solutions to structural design problems provided the design generation is properly constrained. When these constraints are removed, shape annealing generates functional yet spatially innovative solutions for the same design problem (Shea et al., 2005; Shea and Cagan, 1997).

In addition, shape grammars have been combined with cellular automata. The aim of this combination was to explore the potential for reducing the manual effort associated with both cellular automata and shape grammars. Cellular automata comprise rules for evolving the state of a discrete dynamical system. The same rule is applied repeatedly to a system state, and new states are generated in parallel as a step function (Speller et al., 2007). A summary of computational methods combined with shape grammar within research is provided in Table 22 below.

Table 22. Computational methods combined with shape grammars within research.

Method	Finding
Ontology	Ontology and shape grammar may have complementary properties that when combined can strengthen design process.
Graph grammars	The interplay between topology and form can be explored and represented..
Evolutionary algorithms	New superior generations of shape grammars can be evolved. Effort and time can be reduced.
Simulated annealing	Alternatives can be generated for designers with different purposes, while providing insight into relations between form and function.
Cellular automata	The human intuitive approach for visualization of the abstract can be combined with a generative computational system.
User interfaces	Interaction scenarios could for users of shape grammars could include: full control; partial control; no control.

Shape grammars have also been combined with a variety of user interfaces. Perhaps the most sophisticated is an interface that facilitates the building of 3D CAD representations of a product type by starting with inputting words for describing the image of the required product (Hsiao and Chen, 1997; Hsiao and Wang, 1998) It has been proposed that possible interaction scenarios could include: full control; partial control; no control. With full control, the user is responsible for each rule invocation as well as grammar development. This is analogous to a non-computerized system. With partial control, there is user selection during some aspects of rule invocation. With no control, the grammar is predefined, possibly computer-generated. The system automatically generates designs without user intervention. It has been argued that non-modal interactions techniques are preferable because designers work in a non-linear manner as they switch between tasks (Chase, 2002). Further, it has been argued that software demonstrators are necessary for the development of intuitive, interactive interfaces (Shea and Gourtovaia, 2004).

6. Challenges and resources

6.1 Overview

As explained in section 2, the opportunities for sustainable product creation cannot be fulfilled by further implementation of established manufacturing materials and machines; conventional CAD software; and established human design / engineering skills. Accordingly, it will become increasingly necessary to overcome whatever challenges to the development of Generative Production Systems are remaining. Shape grammars provide a good basis for examining for these challenges, because advances in their formulation and computation has been reported in scientific periodicals for three decades. Moreover, the reporting of the development of shape grammars includes numerous criticisms (e.g. Caldas, 2008) and descriptions of challenges to implementation (e.g. Benros and Duarte, 2009). In this section, details are provided of challenges and resources for the development of Generative Production Systems. First, existing implementations of shape grammars are discussed. Then, technological challenges are described under the following four headings: inherent ambiguity, domain complexity, computational complexity, and software development. Next, technological resources are described under the same four headings. In conclusion, an assessment of development opportunities presented.

6.2 Implementations in industry

At least one new proprietary technology which can enable generative CAD design is reported to becoming available. The term, generative, has already been incorporated into the titles of existing CAD software packages used for building design (e.g. Generative Components) and product design (e.g. Generative Shape Design). Despite their use of the term, generative, the potential of such software packages to enable infinite spatial emergence may be quite restricted (Cardoso and Sass, 2008). Moreover, such software packages are for use by professional designers. Thus, they have limited

potential to enable the broadest possible range of populations to carry out sustainable product creation. Also, scripting of partial grammars to enable automated evolution of some geometric features is becoming established in building architecture. However, such scripting is more often intended to enable form-finding rather than form-generation. In form-finding, a geometric form is defined within CAD software possibly after having been generated through traditional design practices which may include hand sketching and manual modelling. Then, the form is evolved through shape deformation enabled by the scripts (Ceccato, 2009). Again, this is the work of professional designers. Moreover, it is often carried out by professional designers who have interest and expertise in computation.

There is one example of an entire shape grammar being implemented in industry. That is a shape grammar to route systems tubing through an airplane. The associated design representations include geometric models (boundary representation solid models), parts and assemblies, part classifications, part interfaces, and functional schematics. The shape grammar is used interactively to generate complete CAD geometry of aircraft tubing, including the associated fittings, clamps, and mounting brackets. The rules include the manufacturing constraints for the geometry of the tubes; constructive rules for tube routing; fittings types and compatibilities; materials; and clearance constraints between different components and systems. The shape grammar was used to design several hundred tube assemblies on the 767-400ER (Antonsson and Cagan, 2001). The development path for this shape grammar was quite conventional. It involved a doctoral scholar carrying his research forward into industry (Heisserman, 1991; 1994; Heisserman and Callahan, 1996). This suggests that the computation of shape grammars can be transferred from research to practice if there is a necessity for implementation.

The lack of industry examples suggests that there may have been very few circumstances which would necessitate implementation of an entire shape grammar for one specific application. However, necessity for an innovation to be implemented often does not occur for years, or even decades. For example, when 19th century chemists distilled petroleum to obtain fuel for oil lamps, they discarded the most volatile fraction, gasoline, as an unfortunate waste product. This practice continued until it was found to be an ideal fuel for internal combustion engines. However, motor vehicles powered by internal combustion engines did not come into widespread use immediately after Gottlieb Daimler built his first engine in the 1880s. This was because, at that time, there was no shortage of horses and no dissatisfaction with railroads. It was not until the First World War that implementation of internal combustion engines, and use of gasoline to fuel them, became a pressing necessity when lorries and horses could not take sufficient supplies quickly enough from railheads to troop positions. Thus, necessity can be the mother of implementation; as well as the mother of invention. With regard to shape

computation there are at least three factors which prevent their widespread implementation before there are circumstances which make their implementation a pressing necessity. These factors are: functionality of conventional CAD software; prevalence of established manufacturing materials and machines; interests of designers and engineers.

Here, the term, conventional CAD software, means CAD software that does not involve shape grammars. Conventional CAD software includes: drafting CAD, parametric CAD, and configurator CAD. Parametric CAD software allows users to define geometrical entities by establishing relationships of geometrical or mathematical dependency between different elements of the design. Once the constraints and chains of dependency are defined by the designer, the driving geometry can be changed in order to produce design variations that retain the logic of the constraints (Cardoso and Sass, 2008). By contrast, drafting CAD software can be regarded as "traditional" CAD tools because they can be seen as evolved, automated versions of traditional drafting table of the architect or engineer. When compared to parametric CAD software, drafting tools are primarily non-hierarchical and non-relational (Cardoso and Sass, 2008). CAD drafting software is prevalent throughout industries. Further, designers from building architects to industrial designers are likely to have been trained with CAD drafting software. Furthermore, parametric CAD software tools are often extensions of CAD drafting software and/or have the similar interfaces and displays as CAD drafting software. Similarly, CAD configuration tools are often extensions of CAD drafting software and/or have the similar interfaces and displays as CAD drafting software. CAD configuration tools are used in conjunction with some assemble-to-order (ATO) market offerings such as coaches; conservatories; elevators. CAD configuration tools (i.e. configurators) are software tools which can help users, including customers, to develop approved product configurations quickly and accurately with a minimum of effort. Users create designs by picking from pre-determined component options and arranging them to suit their own individual requirements. Often, CAD configurators can generate information for manufacturing such as bills of materials or cutting lists. The prevalence of established manufacturing materials and machines in product creation means that best available material / machine utilizations can be incorporated into conventional CAD/CAM software, as well as into CAD configurator software. This can be achieve through, for example, data validation routines; libraries of parts; parametric product models.

It is possible that shape grammars could be implemented to improve the functionality of conventional CAD software and CAM software in meeting established design challenges. However, diffusion of innovation requires awareness of the innovation among potential implementers (Larsen, 2005; Rogers, 2003), and awareness of shape grammars may be limited because research into shape grammar tends refer only to itself

(Gerzso, 2003). Moreover, improving the functionality of conventional CAD software with shape grammars could involve there being less employment for professional designers. This could happen, for example, if shape grammars were implemented to automatically generate more design options than several professional designers could generate. Similarly, improving the functionality of conventional CAM software could involve there being less work for professional engineers. Accordingly, professional designers and professional engineers could perceive the implementation of shape grammars as being not in their interests. Accordingly, it is possible that professional designers/engineers might resist implementation of shape grammars even if they did know about them.

It is notable that the one known shape grammar implementation in industry is being used to carry out design work that offers few opportunities for human designers / engineers to exercise professional flair from which they are likely to receive a high degree of internal satisfaction or external recognition. Although important, the routing of systems tubing through an airplane is not a particularly prestigious design / engineering assignment. Indeed, it is an assignment that human designers might prefer to have automated. Moreover, it is an assignment that can clearly benefit from the application of generative geometric design enabled by shape grammars. This is because it is unlikely that the best available solution can be obtained and verified without the generation of many alterative routings. Further, airplane tubing is well suited to part count reduction, assembly simplification and weight reduction through the application of advanced manufacturing and materials. This means that prior knowledge of established manufacturing materials and machines is of limited usefulness. Hence, it can be argued that the functionality of conventional CAD software; the prevalence of established manufacturing materials and machines; the interests of designers and engineers may not be major barriers to the implementation of shape grammars for the routing of systems tubing through an airplane. Moreover, as explained in section 2, the opportunities for sustainable product creation cannot be met by further implementation of established human design / engineering skills; conventional CAD software; and established manufacturing materials and machines. Accordingly, it will become increasingly necessary to overcome whatever challenges to shape computation implementation are remaining from research. Shape grammars provide a particularly good basis for examining for these challenges because the development of their formulation and computation has been reported in scientific periodicals for three decades. This reporting of the development of shape grammars continues to include numerous criticisms (Caldas, 2008) and descriptions of challenges to implementation (Benros and Duarte, 2009). Technological challenges are discussed in the next section; before technological resources are described in the subsequent section.

6.3 Technological challenges

Challenge 1: Inherent ambiguity

The computation of shape grammars has to be able to deal with the ambiguity that is inherent within design requirements and the designs that are generated in response to them. It is claimed that in design, ambiguity serves a positive and deliberate function (Fleisher, 1992). In principle, shape grammars can be devised to take advantage of ambiguity in creating novel designs. However, ambiguity, in general, is inherently counter-computable, and the level of ambiguity has to be controlled for any computational implementation to be tractable. In order to translate into programming code, shape rules have to be specified in a computational-friendly way: that is, shape grammars need to be quantitatively specified and there needs to be enough precision in the specification to disallow generation of ill-dimensioned configurations.

As well as the ambiguity that is inherent within design requirements and the designs that are generated in response to them, shapes themselves can be ambiguous (Knight, 2003b). For example, the shape of a circle in an architectural drawing can be part of a sign indicating north; a round stool in plan, a manhole cover, a lighting fixture, a decorative pattern, and so on. Within natural language, there is also great potential for vagueness and ambiguity. For example, the word, cat, can refer to a feline mammal; a youth; or a computed axial tomography scan. However, semantics supplies meanings to sentences, and these meanings may not change as the composition accumulates. In contrast to reading, seeing compositions of shapes is an act of renewable and revisable organization. In particular, the meaning of shape cannot always be the accumulation of the meanings of sub-shapes (Fleisher, 1992). Further, natural language sentences have a linear sequence, while shapes can be three dimensional. Also, compositions of shapes are realized through the physical processing of materials and physical experiences of people.

Such factors call into question the robustness of Chomsky's formal grammars as a basis for shape grammars. Further, there is debate as to what type of grammar is most relevant to shape grammars. For example, it has been argued that designs are very context sensitive, and that there is always an exception to the rule (Rudolph, 2006). Thus, the relative merits of Chomsky's context free grammars, Chomsky's transformation rules; and attribute grammars are discussed in the research literature (Fleisher, 1992; Schmittwilken et al., 2007). Moreover, it has been argued that some uses of labels and markers in shape grammars can make them ambiguous grammars (Aho and Ullman, 1972). Further, it has been argued that insufficient understanding of grammars is a significant flaw within shape grammar research. In particular, it has been pointed out that papers on shape grammars tend to only refer to other papers in their own field, and fail to convincingly relate their work to the vast literature on grammars in other fields (Gerzso, 2003).

Challenge 2: Domain complexity

Some challenges for shape grammar implementation within the domain of engineering have been defined as listed below (Deak et al., 2006).

- Engineering domains have a large set of inherent domain requirements, and each specific design to be generated will have a large set of problem specific requirements and constraints related to that instance. Creating a grammar rule set that contains the maximal amount of domain knowledge, while remaining flexible and adaptable enough to fulfil the greatest number of designs can result in a large complicated grammar rule set.
- Further, in order to use shape grammars in an automatic design generation scenario in most engineering domains, the grammar has to prohibit the introduction of flaws into the design.
- Yet, it is very difficult to verify a grammar. A recursive rule set can define an infinite space of possible solutions, and therefore contain designs that may be flawed in ways that were not anticipated by the grammar designer.
- It is difficult to create a "designerly" grammar, where the order and application of rules proceeds in a way that makes sense to the user.
- Communicating grammar is difficult; justification for individual grammar rules can be difficult to provide, as they may not have a direct significance on a design, instead playing a linking role where they prepare parts of the design for further grammar rules to work on. This can make maintenance, and understanding of the grammar by anyone who was not involved with its creation difficult.

Challenge 3: Computational complexity

Typically, high-level programming skills are required for the production of sophisticated and highly detailed designs using shape grammars (Sass and Oxman, 2006). Challenges for shape grammar computation have been defined as listed below (Chau et al., 2004):

- incorporate intuitive user interface
- enable subshape recognition and shape emergence
- enable automatic shape recognition under Euclidean transformations
- allow parametric shape rules
- enable automatic shape recognition for parametric shape grammars
- allow three dimensional shapes
- allow curvilinear basic elements
- support surfaces and solids
- provide aesthetic measures for ranking designs for automated selection

- provide unambiguous interpretation of resulting designs to their physical realization.

More broadly, it has been argued that improving the reasoning abilities of computational tools will continue to be a challenge (Cagan et al., 2005), and that parametric design is preferable to shape grammar computation when the level of intelligence of the desired program is low (Benros and Duarte, 2009).

Shape grammar schema may be necessary to address two important challenges. First, designs can be grouped into categories on the basis that they are dimensional variations of each other. However, shape grammars require that each member of a category be computed individually. Since categories can be indenumerably infinite, so can computations over their members. Accordingly, it would be very useful to be able to apply grammar rules to entire categories at once. Second, shapes are often used in application to represent specific geometric constructs. Notions of incompleteness in grammatical derivation must be dealt with through addition of new parts of a shape or modifications of existing parts. There is no inherent notion of partiality. Shape schemata are objects that capture a class of shapes by assigning variables to the structure of a shape and giving a set of constraints over the variables. Any assignment of specific values to the variables creates a (possibly reducible) shape and any such assignment that is consistent with the constraints of a schema is an instance of the schema (Woodbury, 1997).

Challenge 4: Software development

Overall, there is a need for shape rules to follow a certain computation-friendly framework. There has been some success in efforts to reduce the coding work required to develop shape grammar interpreters with increased functionality. For example, counter-computational hindrances that commonly occur in shape grammars have been addressed (Yue and Krishnamuri, 2008). Also, reductions in computational load in the communication of shape grammar have been achieved (Sass, 2007b). Further, with regard to the combining of shape grammars with optimization techniques, the A-Design agent-based approach has been used to reduce the level of stochastic search (McCormack and Cagan, 2002). Nonetheless, in spite of the depth of shape grammar research, there are no robust general purpose parametric interpreters for shape grammars. This deficiency has limited the practical use of shape grammar to hand manipulation, which serves an educational value in itself, and by self-development of customized software which may have limited generalizability for other applications (Stiny quoted in Speller et al., 2007). This may be due to mathematical complexity (Piazzalunga and Fitzhorn, 1998) and the potential for shape grammars to produce a possibly infinite number of outcomes (Knight, 1996).

On the other hand, shape grammar interpreters have undergone considerable progression within research. Since the early 1980s, for example, interpreters have progressed from maximal representation of straight lines; to three dimensional geometry; to complex solid models; to shape grammar schema; to parametric shape recognition; and to shape emergence (Agarwal and Cagan, 1998; Flemming, 1987; Heisserman, 1991; 1994; Krishnamurti, 1982; Krishnamurti and Giraud, 1986; Piazzalunga and Fitzhorn, 1998; Tapia, 1996; 1999; Woodbury, 1997). Accordingly, it is quite possible that the deployment of resources by industry could lead to further progression. With regard to the development of user interfaces, the interactions available to users can be limited in scope when compared to conventional CAD software. This may be due to a lack of understanding about how grammars relate to the design process to the difficulty of handling the unexpected nature of emergent features (Chase, 2002). Indeed, it has been opined that alternatives to shape grammars are required to better enable the emergence of unexpected design characteristics (Caldas, 2008). A summary of challenges to implementations of shape grammars is provided in Table 23 below.

Table 23. Shape computation challenges.

Challenge	Example
Inherent ambiguity	One shape can have several meanings.
Domain complexity	Large sets of problem specific requirements and constraints.
Computation complexity	Automatic subshape recognition.
Software development	Difficulty of handling emergent features.

6.4 Technological resources

Resources for dealing with inherent ambiguity (Challenge 1)
Although it is important to draw attention to differences between reading natural language (linear sequence) and seeing compositions of shapes (three dimensional arrangements); it is also important to recognize that natural language discourses which are handled by computation can be far from being linear. Rather, they can be iteratively multimodal and involve prosody, gesture, reiteration, etc. Furthermore, gestures which are often used instead of words, such as pointing, take place in three dimensions. The challenge of computing ambiguous meanings in natural languages, within implementations such as in car spoken dialogue systems and intelligent tutoring systems, is dealt with by computational semantics. An introduction to computational

semantics is provided by Blackburn and Bos (2005). Implementations of computational semantics actually involve integrated computation of grammar, pragmatics, and machine control, as well as semantics. Often these are combined in mixed initiative systems which enable the lead in human-computer discourse to be taken either by the human or the computer as necessary. By contrast, the need for mixed initiative discourses in design space exploration is recognized within research, but has not been implemented (Datta, 2006). Further, implementations of computational semantics involve intuitively understandable user-interfaces which have yet to be developed for shape computation.

Like shape grammars, computational semantics draws upon Chomsky's grammars. For example, in the analysis of spoken input into an in-car dialogue system such as, "where can I get Chinese?" (i.e. where is the nearest Chinese restaurant?). However, it is important to note that different grammars are used within computational semantics. Chomsky's Context-Free Grammar formalism may be used. Alternatively, Unification Grammar may be used. The advantage of context-free grammar is that it provides a simple and precise mechanism for describing the methods by which natural language phrases are built from simpler components. The disadvantage is that important features of natural language such as agreement and reference cannot be easily expressed. The advantage of Unification Grammar is the potential to encompass multimodal inputs, such as gesture, as well as natural language.

Also, while it has been argued that semantics supplies meanings to sentences in natural language, and these meanings may not change as the composition accumulates; it is important to note that understanding of what is meant can certainly change as the composition accumulates. For example, the understanding of the sentence, "I shall walk to the bank..." can change as the composition accumulates "...to collect my new credit card" or accumulates "...to see the new jetty onto the river". Thus, the argument that: in contrast to reading, seeing compositions of shapes is an act of renewable and revisable organization, is somewhat superficial. This is because each reader's understanding of written words can be an individual act of renewable and revisable organization. Further, different participants' understanding in a natural language discourse can often be renewed and revised in accordance with their perceptions of congruence or incongruence between the words that people utter and their tones of voice. Perceptions of a tone being sarcastic, for example, can lead to a sentence such as "you have done a good job once again, I see" being understood to be critical. Then, when the discourse proceeds to the silent handing over of a bonus payment, the initial perception of the statement can be revised. Moreover, while it has been argued that the meaning of shape cannot always be the accumulation of the meanings of sub-shapes; it is important to note that the meaning of natural language discourses cannot always be the accumulation of the meaning of each dialogue act. Rather, some dialogue acts in a discourse mean

more than others. One punch, for example, can carry far greater meaning than a thousand conciliatory words that are spoken before it is thrown. Thus, in natural language discourse, as in the combinations of shapes, proportion can exert a determining influence over meaning. The potential for evolution of meaning as composition accumulates, and discourses progress, is dealt with in the computation of natural language through the use of underspecified semantic representations. These representations are assigned to dialogue acts, such as speech and gesture, and allow for further specification as composition accumulations, for example through the progression of discourse.

Further, while it is important to note that compositions of shapes are realized through physical processing of materials (e.g. to make products and places) and human experience (e.g. holding a product; walking in a place), it is also important to recognize that natural language interactions that are handled by computational semantics implementations can also involve physical processing and human experience. For example, in car spoken dialogue system for music selection will control machines such as DVD player, speakers etc. From this machine control comes computer output such as the playing of requested music. Alternatively, computer output could be a spoken query through a speech synthesizer.

Overall, it can be argued that the computation of shape could benefit from application of techniques used in the computation of multimodal natural language interactions.

Resources for dealing with domain complexity (Challenge 2)

It is important to draw attention to domain complexity. For example, that each specific design to be generated will have a large set of problem specific requirements and constraints related to that instance. However, it is also important to recognize that domain complexity is not an insurmountable barrier to implementation. For example, successful computational semantics applications, such as Intelligent Tutoring Systems (I.T.S.), operate within the complexity of domains such as ship handling. I.T.S. use simulations and other highly interactive learning environments that require trainees to apply their knowledge and skills within the context of a particular domain. I.T.S. provide individualized guidance by developing a model of each trainee's skills via observation and assessment of their actions within interactive environments. I.T.S. can, for example, infer trainees' level of confidence through recognition of vocal cues such as rate, pausing and pitch. Such cues might also be used to detect conceptual misalignment, when a trainee's understanding of a concept is different from that of the tutor. Thus, Intelligent Tutoring Systems deal with the complexity of instructional domains, such as ship handling, and at the same time deal with the complexity of human–computer discourse.

This compound complexity is dealt with in computational semantics applications by what can be thought of as a "team of experts". The team of experts is not a team of human experts but a team of computational tools. These include grammars, but can also include domain ontology; communicative ontology; Stochastic Language Models; Support Vector Machines. These different computational tools can have overlapping expertise, and each can be more or less useful within the same application depending on the nature of different communications that each application encounters. Thus, developers of computational semantics applications recognize that not all functional demands can be met solely grammars. Rather, they recognize that all grammars "leak" to some extent. In other words, no grammar is capable of dealing with every aspect of every communication that an application may encounter. Within the field of shape grammars, by contrast, there may be insufficient recognition of the advantage that can arise from placing grammars within a team of computational tools.

Resources for dealing with computational complexity (Challenge 3)

It is important to draw attention to computational complexity. For example, to draw attention to the computational complexity of achieving robust automatic subshape recognition. However, it is also important to recognize that computational complexity is not an insurmountable barrier to implementation. For example, computational semantics applications, such as in-car dialogue systems, operate successfully within domain complexity and human-computer discourse complexity through use of computational tools such as Stochastic Language Models or, so called, classifiers including Support Vector Machines. Stochastic Language Models (SLM) can process inexact inputs such as mispronounced or misspelt words. SLM assign a most probable classification to an input based on a probability developed through observation of, for example, the language used by domain experts in their work. Classifiers, such as Support Vector Machines (SVM), use knowledge of features to determine what something is. SVM make use of pattern recognition algorithms. The application of classifiers may have the potential to overcome the need to have expert programmers add new unanticipated parts to shape grammars that are found to be insufficient by users. For example, a user of a shape grammar could draw an additional shape and it could be processed by a classifier. The use of digital pens could make this a fully computational process.

Also, the challenges faced by expert human programmers can be reduced through machine learning. The term, machine learning, refers to the ability of a program to learn from experience. In other words, to modify its execution on the basis of newly acquired information. Machine learning techniques include Markov models; neural networks; decision tree classification; and vector-based clustering. Within computational semantics, machine learning techniques have been applied to classification, to parsing, and to dialogue management. All of these can involve applying machine learning

algorithms to corpora. The term, corpora, is used to refer to sets of existing texts. It can also be used to refer to existing designs. Machine learning techniques depend on having some initial "training examples" prepared by human experts to learn from. Nonetheless, the use of machine learning can greatly reduce the need for human expertise in corpora analysis and in subsequent programming work. Within computational semantics applications, machine learning can be restricted to the development phases of computational semantics applications. Alternatively, machine learning techniques can be deployed to enable an application to learn from new instances during its use. This is can be referred to as "learning in the wild". The potential benefits of applying machine learning to shape computation have been recognized for some years (Gero et al., 1994), but have yet to be widely applied. In efforts to reduce the computational challenges faced by individual expert human programmers, a potential alternative, or complement, to machine learning could be human-based computation. This can involve having large distributed groups of non-expert people working on small components of a computational challenge. This can be achieved by setting up on-line games that extract knowledge from people, with their consent, in an entertaining way.

Resources for dealing with software development (Challenge 4)

It is important to draw attention to the general lack of progress in developing user-friendly software packages for shape grammars. However, it is also important to recognize that seeking to develop general purpose parametric interpreters may be counter-productive. Given the domain complexity and computational complexity outlined above, specialization of software packages may lead to the more rapid development of more robust programmes. By contrast, endeavouring to develop a software program that is good for everything could lead to the development of software that is not very good at anything. In particular, breadth of scope could come at the loss of accuracy in domain.

Nonetheless, there will be activities which are common to the development of specialized software packages. Across a range of computational semantics applications, for example, the early stages of development often involves typical systems analysis activities. In particular, developers have to learn about the real world domain of interest. Then, for example, when developing an Intelligent Tutoring application, the developers will need to learn about general strategies for tutoring. Further, the developers will need to learn about the specific application domain. For example, when developing an Intelligent Tutor for ship handling, the developers will need to learn about the many aspects of seamanship that are relevant to ship handling. Furthermore, the developers will need to learn about the communication procedures, jargon, prosody etc., which are prevalent in ship handling. In addition, the developers will need to learn about the working of devices that will be used in the application such as devices for speech

recognition, speech synthesis, visualization etc. Also, the developers will model the system, and component devices, that will be interacted with during that activity. Moreover, the developers will model how communication processes will embody the interactions involved in that activity. All together this enables the formulation of preliminary systems architecture for the particular application. Then, applications can be developed through configurable modular systems architectures. This enables initial components to be developed concurrently by different contributors. Also, new components can be added as the scope of an application develops. Moreover, components can be enhanced over time as computational semantics advances and/or greater resources become available.

Alignment of shape computation programs with legacy CAD / CAM systems could be an important issue when, for example, the costs of transferring all of production to AMM are initially too high for companies in the value chains for consumer goods. In such cases, examples of alignment among computer languages during computational semantics applications can provide useful insights (Sowa, 2008).

Overall, it is important to note that fewer programming resources have been available for developing shape grammar software than for developing, for example, computational semantics application software. The development of robust user-friendly software packages may be much more likely when increased programming resources are combined with a focus on increased specialization, and the deployment of modular systems architectures. Moreover, robustness may be facilitated by determination of which activities can be handled more reliably outside of shape grammars and considered extragrammatical or noncomputational. A summary of resources available for meeting implementation challenges is provided in Table 24 below.

Table 24. Shape computation resources.

Challenge	**Resources**
Inherent ambiguity	Techniques used in multimodal natural language computation
Domain complexity	"Team" of integrated computational tools such as SLM, SVM
Computational complexity	Classifiers; machine learning; human-based computation
Software development	Specialization; configurable modular systems architectures

6.5 Assessment

The consideration of resources for meeting implementation challenges has made extensive reference to computational semantics applications. Within shape grammar research, there has been little consideration of computational semantics. One possible example of a starting point is an image database that has been developed for relationships between product shapes and image words. This makes it possible to initiate and regulate product shapes by inputting image words (Hsiao and Chen, 1997; Hsiao and Wang, 1998). More broadly, the need to consider semantics has been discussed within shape annealing (Cagan and Shea, 1999) and graph-based grammars (Rudolph, 2006). It has argued, for example, that through the use of labels, semantics based on functional, behavioural, and aesthetic design requirements can be incorporated into a shape grammar. This creates a functional grammar that enables the quantification of relations between form and function in the spatial layouts of functional systems (Shea and Cagan, 1997). However, as illustrated in Figure 8, the thinking, methods, and techniques, that have been used together to enable successful computational semantics applications have yet to be applied to shape computation.

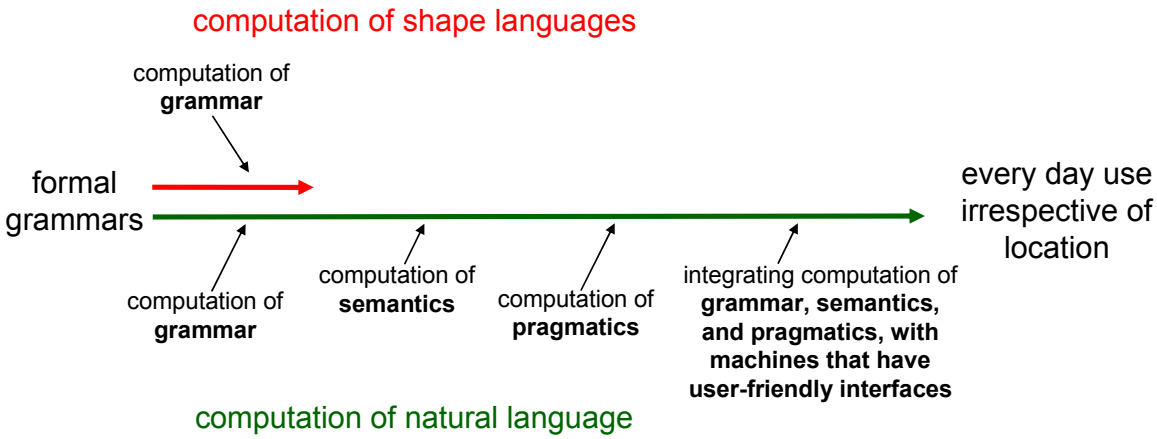


Figure 8. Computation of shape language.

Thus, the combination of computational semantics with shape grammars offers a potentially fruitful direction for the development of the Generative Production Systems needed for sustainable production creation. Potential fields of application for Generative Production Systems are introduced in the next section.

7. Examples of application opportunities

7.1 Overview

In this section, examples of application opportunities for Generative Production Systems are provided. To begin with, opportunities which are common across economic sectors are outlined. Subsequently, more specific opportunities are discussed. First, the potential for Generative Production Systems to enable integration of elicitation with design and production is discussed. Second, the potential for new business models arising from use of Generative Production Systems by non-experts is considered. Then, opportunities for more rapid exploration of materials' potential, and of parts consolidation, are examined. Next, opportunities for rapid generation of new product/component styles, and building environment topologies/geometries, are outlined. Subsequently, opportunities for creation of different types of products are defined. In conclusion, preliminary criteria for generative production criteria are provided.

7.2 Integration of elicitation with design and production

Established business models for product creation involve design being carried out by professional experts in disciplines such as architectural design; heating, ventilation, and air conditioning design; industrial design; engineering design; design for manufacturing and assembly; etc. Design experts from different disciplines may be located together at one place in one company or working at different global locations for different companies. Also within established business models, production is carried out by professional experts in fields such as manufacture, fabrication, assembly, and installation. All of production may be carried out at a single location or may be carried out at different global locations. Production experts contributing to a product may all work for one company or may all work for different companies. As shown in Figure 9,

established product creation business models have a variety of titles but can be categorized in four groups.

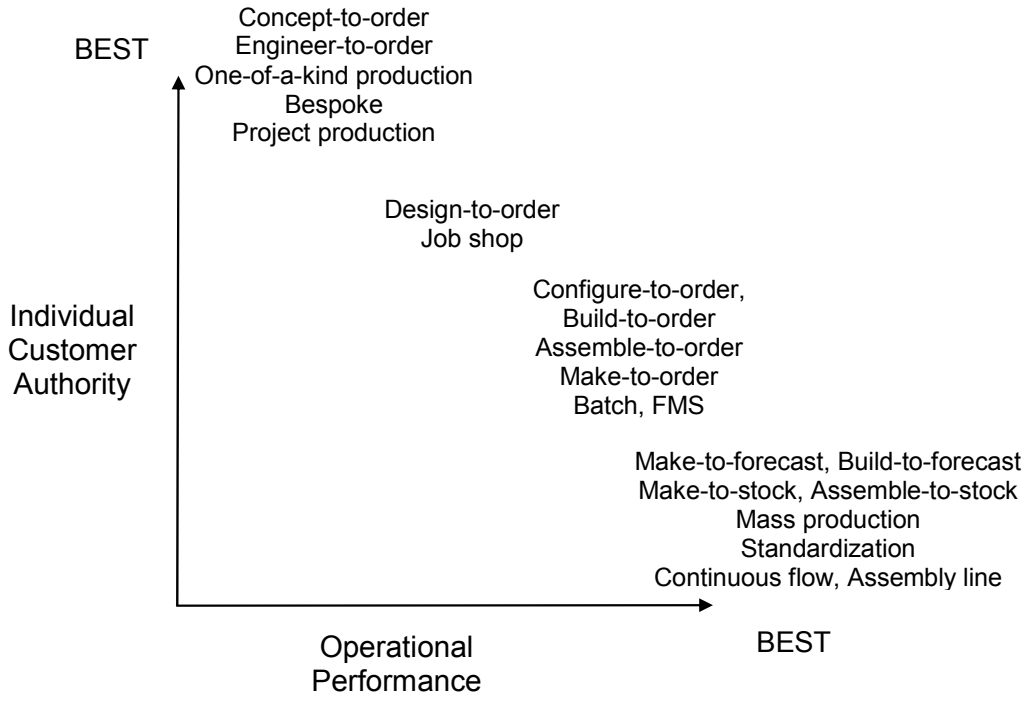


Figure 9. Established business models for product creation.

These four groups comprise a continuum (engineer-to-order; design-to-order; assemble-to-order; make-to-stock). As shown in Figure 9 and listed below, different business models involve offering different types of authority to individual customers. Typically, companies achieve best operational performance (e.g. highest productivity, lowest defects) when they make standard goods to stock.

- Engineer-to-order: authority over design and production (e.g. products such as buildings and ships)
- Design-to-order: authority over design (e.g. products such as wedding dresses)
- Assemble-to-order: authority over selection and configuration of pre-designed components (e.g. products such as cars)
- Make-to-stock: authority over selection of pre-designed products (e.g. fast moving consumer goods).

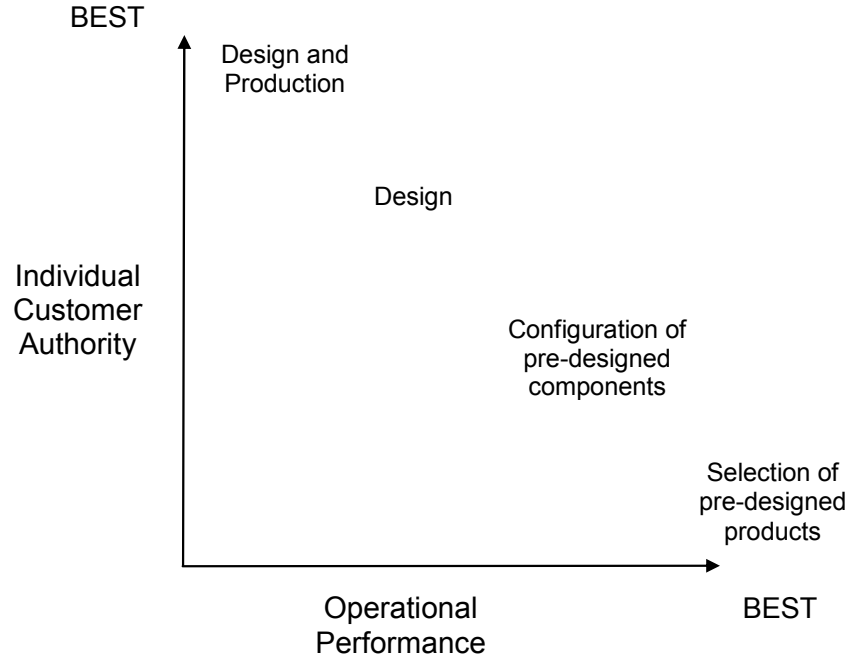


Figure 10. Established business models: customer authority.

As shown in Figure 11 below, different business models encompass different levels of involvement from individual customers during the elicitation of requirements. In particular, the engineer-to-order and design-to-order business models involve elicitation of actual customers' order-specific requirements. By contrast, assemble-to-order and make-to-stock involve various methods for eliciting the opinions and ideas from samples of potential customers within different market segments. These methods include, for example, market research; user-centred design; on-line communities; on-line competitions; etc. It is important to note that although assemble-to-order organizations (e.g. car makers) and make-to-stock organizations (e.g. personal care product companies) have increasingly sought input from users of their products, they continue to offer pre-determined choices, rather than authority, to individual customers.

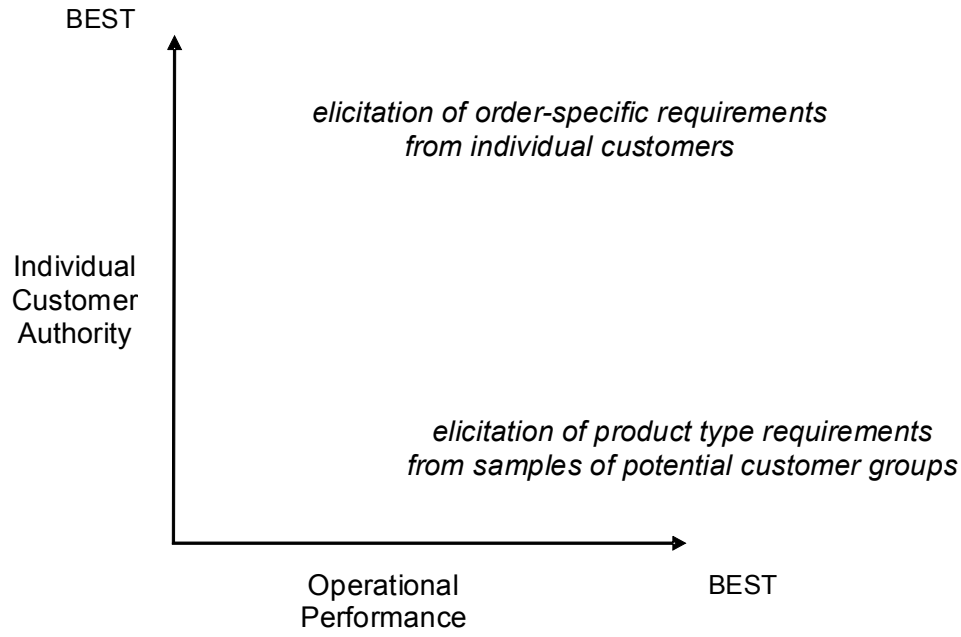


Figure 11. Established business models: elicitation.

As illustrated in Figure 12 below, the elicitation of requirements, the development of designs, and the production of physical goods currently involves numerous languages barriers. These can be barriers among natural languages used by the different disciplines involved and the different types of customers/users of physical goods. Also, these can be barriers among the computer languages used by different types of software in elicitation, design and production. As illustrated in Figure 13, Generative Production Systems which are based on the computation of spatial elements such as points, lines, planes and volumes rather than fixed primitives have the potential to introduce what can be described as unified shape production languages (USPL). Those being shape languages that can be used effectively throughout elicitation, design and production. As discussed in section 5, file formats such as STL, and Markup Languages such as XTML, can enable shapes to be processed by a variety of CAD/CAM systems. In simple terms, USPL should enable seamless progression from mental visualization of an artefact to physical production of that artefact (i.e. from mind to machine). This could do much to enable product creation by non-experts, and so facilitate the meeting of needs, and the expression of potential, that is essential to sustainable product creation.

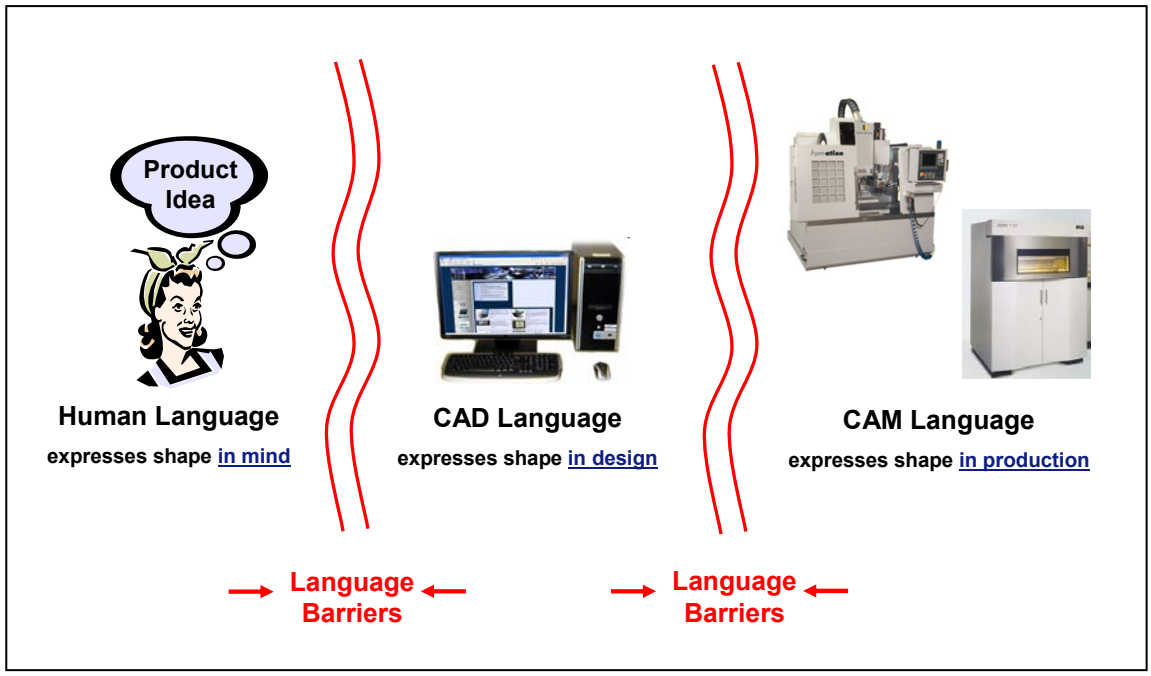


Figure 12. Different shape languages in mind, in design, in production.

Figure 13. Unified Shape Production Language (USPL).

7.3 New business models

As shown in Figure 14 below, beyond existing business models for product creation, it is envisaged that individual customers with little, if any, prior expertise could design and produce sophisticated products themselves by using Generative Production Systems. Moreover, individuals would be able to create the products that they want at times and costs commonly associated with make-to-stock and assemble-to-order products.

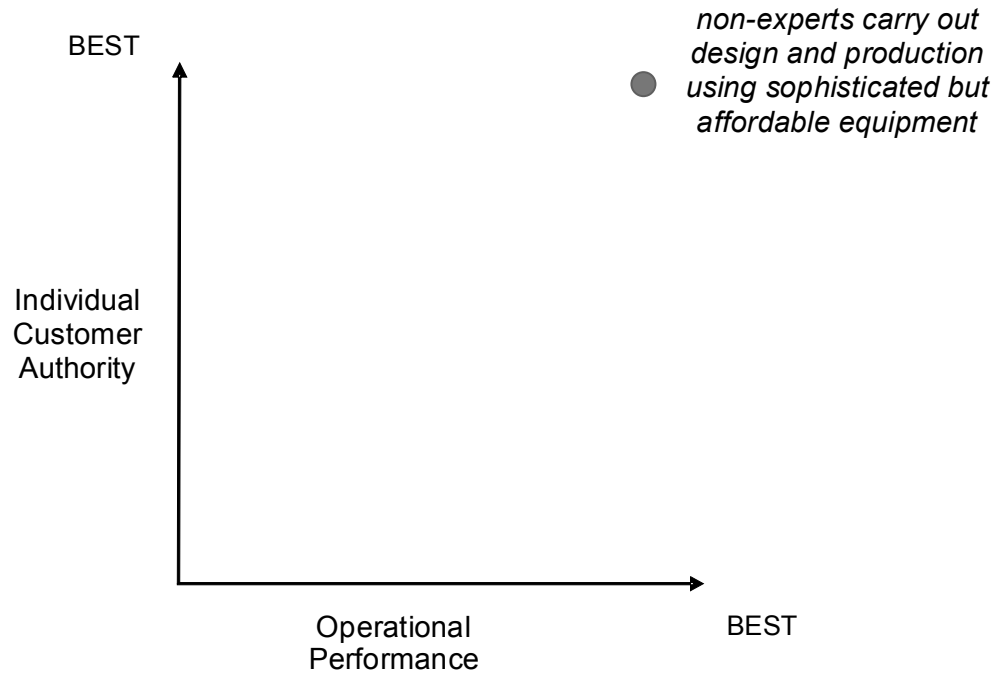


Figure 14. Beyond established business models.

Further, combining Generative Production Systems with Web 2.0 could revitalize manufacturing and generate new employment. Web 2.0 is the second generation of the World Wide Web. It has enabled the movement away from static web pages to creation of dynamic, shareable content by non-experts working individually and/or within Web 2.0 enabled social networks. Already a few companies, such as FigurePrints, Ponoko and Shapeways, have recognized the potential of combining Web 2.0 with AMT. For each customer, FigurePrints takes digital data which describes a character in a virtual game and, then, manufactures a three-dimensional physical replica. In doing so, FigurePrints connects the synthetic economy of virtual world transactions with the real economy of exchanging physical goods for money. Through its website, Ponoko offers

kits to help individual customers design and make a variety of physical goods. Shapeways offers on-line support for individual customers to design and sell products that can be produced by additive layer manufacturing.

More generally, Web 2.0 provides dynamic shareable content to fuel ideation and so lead to many new product ideas. Also, Web-enabled social networks like facebook®, flickr®, youtube® can be used to rapidly propagate new ideas around the world. Until now, established product brand holders, such as Electrolux, Lego, and Philips, have used Web 2.0 to harvest the great product ideas of ordinary people via on-line communities and competitions. They have then fed those great ideas into their design departments. In doing so, they have stuck with the paradigm of concentrated design, production and dispatch that began with the industrial revolution. Such use of Web 2.0 can introduce further complexity into the operations of established brand holders and their suppliers. For example, they can have many more product ideas to evaluate. Next, they can have many more product types that need to be designed, manufactured, and packaged. Alternatively, existing brand holders can choose to discard ninety-nine percent of the product ideas that they harvest via the Web. This can avoid introducing further complexity into their operations but, on the other hand, this can lead to many opportunities for increased sales being missed.

Instead, existing brand holders could harness the full potential of Web 2.0 by allowing people who come up with new product ideas to operate design and production under licence using Generative Production Systems. Some established brand holders already provide web-based design tools. The next step is for them to allow transfer of digital design data to point-of-demand AMTs. In addition to the reduction of environmental impacts, this offers at least four advantages for established brand holders. First, ever more complexity is not introduced into existing design and production. Second, opportunities for potential sales within their brand are not missed. Third, additional demand for core brand products is stimulated by the profusion of new add-on products introduced via Web 2.0 plus point-of-demand Generative Production Systems. Fourth, established brand holders receive a royalty for each sale via Web 2.0 plus Generative Production Systems. The advantage for national economies is that the people who come up with new product ideas can, depending on the amount of sales, make some money, employ themselves, set-up businesses employing others. Also, Web 2.0 plus Generative Production Systems enables work to stay where product ideas originate, rather than work being off-shored by established brand holders. In other words, the three hundred year old paradigm of concentrated design, production and dispatch carried out by experts can be supplemented by the new paradigm of Factory 2.0. As illustrated in Figure 15, Factory 2.0 can be summarized as highly distributed sustainable ideation, propagation and creation of physical products – enabled by Web 2.0 plus Generative Production Systems.

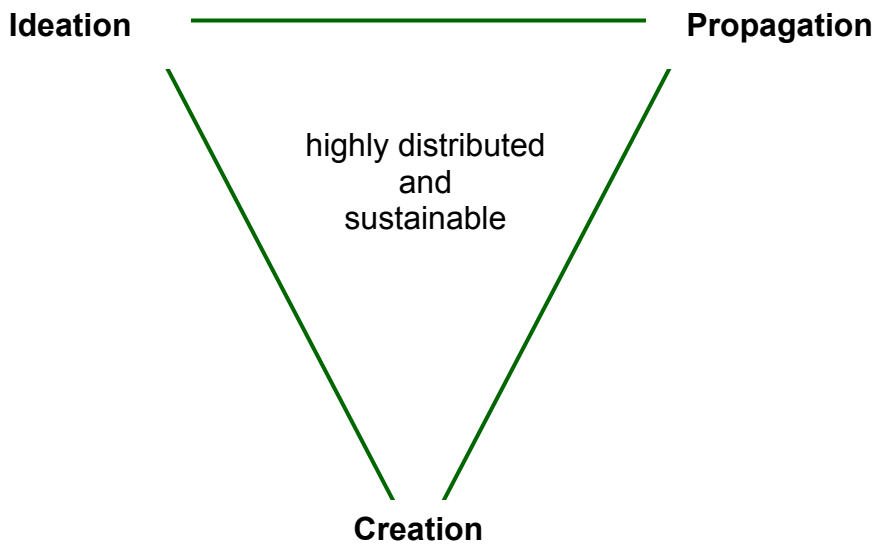


Figure 15. Factory 2.0: Web 2.0 plus Generative Production Systems.

An even bigger opportunity for establishing new employment is for individuals to ideate and propagate products outside of existing brands. Consider, for example, how downloading enabled by Web 2.0 has allowed open digital propagation of new music to supersede closed physical distribution of music. Previously, new music had to pass through the high control gates of brand holders such as record labels. These control gates are high because of the high prior investments of brand holders and their suppliers. By contrast, digital propagation of music via Web 2.0 requires such low investment that almost anybody can offer their musical work for sale. Now, there is no fundamental reason why digital design data cannot be downloaded on demand for physical production to point-of-demand AMTs. As a result, design can be carried out anywhere a product idea originates and production can be carried out at any point of demand. Point being both point on the map and point in time. Moreover, point-of-demand production can include the manufacture of products with a high level of functionality. This can be achieved with so called, "direct-write", additive manufacturing machines which incorporate the functionality of circuits, sensors, controls etc., into assemblies during their production.

The creation of physical products with Generative Production Systems includes their design as well as their production. Accordingly, the full potential of AMTs will be unlocked when non-AMT experts are able to carry out both design and production. As well as people with great new product ideas, non-AMT experts are a wide range of product customers, for example, displaced people who need to make robust low cost housing; medical technicians who need to make prosthetics; maintenance personnel who

need to make replacement fittings; gift shoppers who want to make jewellery; and children who want to make toys. User friendly AMTs, such as 3D printers, are becoming increasingly affordable through the efforts of established machine companies and communities of AMT enthusiasts. Thus, production by non-AMT experts is becoming increasing feasible and viable. To enable highly distributed point-of-demand production, suppliers can install AMT equipment at a variety of locations including wholesaler premises (for B2B sales) and retail outlets (for B2C sales). Production time can be bought on AMT machines and generate additional income for established AMT suppliers. In addition, when there are enough sales, new businesses based on new Factory 2.0 products can buy or lease their own AMT machines and locate them at points-of-demand. A summary of the potential advantages of Factory 2.0 is provided in Table 25 below.

Table 25. Advantages of Factory 2.0.

Type	**Example**
Existing brand holders	Further complexity is not introduced into existing design and production. Opportunity for potential new sales lines within their brands are not missed. Additional demand for core brand products is stimulated by the profusion of new add-on products introduced via Factory 2.0. Brand holder receives a royalty for each Factory 2.0 sale within existing brand.
Eco-systems	The potential environmental advantages of AMT, which are summarized in Table 1, can be realized.
National economies	The possibilities for AMT to enable the expression of greatest potential, as summarized in Table 2, can be realized. The possibilities for AMT to enable the meeting of needs of societies, as summarized in Table 3 can be realized.

In conclusion, it is important to recognize that the established paradigm of concentrated design, production and dispatch by experts will continue to be useful for some types of products. Nonetheless, the established paradigm has limited potential to minimize environmental impacts, and to generate new regional employment. By contrast, Factory 2.0 can create new jobs anywhere that product ideas originate – and enable much lower consumption of materials and energy wherever there is demand for those products.

7.4 Rapid exploration of materials' potential

When the use of a material is well established in product creation, its potential is usually well defined in professional education, and in CAD/CAM software. Its geometric limits, mechanical characteristic, manufacturing properties, etc., for example, are well known and widely applied. The design, engineering, manufacturing and assembly potential of some synthetic materials, such as polymers and composites, have been defined over decades of experimentation. The potential of many more synthetic materials, such as ceramics and alloys, have been defined over centuries of trial, error and experimentation. The potential of natural materials, such as wood and stone, have been defined over millennia of trial, error and experimentation. Further, new materials are often modifications and/or combinations of established materials. Hence, variations in geometric limitations, mechanical characteristics, manufacturing properties, etc., can be quite predictable and easy to test within established experimental procedures. Moreover, new materials are very often applied within established conventions for design, engineering, manufacturing and/or assembly. Such new materials provide professional experts with new possibilities for creating improved products. These could be, for example, smaller and/or lighter consumer products such as mobile phones. Also, they could be, for example, wider and/or taller, built products such as bridges.

Accordingly, there is only need for rapid exploration of materials' potential when that material needs to be introduced quickly and that material (i) will not be applied within established conventions for design, engineering, manufacturing, assembly; and/or (ii) that material is not known to be similar to an established material. The increasing need for sustainability will accelerate the need for rapid introduction of materials that will not be applied within established conventions. Manufacturing with physical 3D voxels is one important case of materials not being applied within established conventions (Hiller and Lipson, 2009). The term, voxel, is an abbreviation for "volume element", and can be thought of as being something like a three dimensional version of a pixel. Physical voxels can be used as building blocks for additive layer manufacturing. In particular, additive layer manufacturing involving selective arrangement of voxels in a three dimensional lattice. This discontinuous (i.e. digital) material placement is a very important new alternative to the continuous (i.e. analog) material placement that is typically used in manufacturing. It is very important because it opens up the possibility for both design and production to comprise the same digital building blocks: voxels that have virtual representation in design, then physical representation in production. This, combined with the potential of unified shape production languages to integrate the elicitation of requirements with design, opens up the possibility of a shortened and fully digital process of product creation. This possibility is illustrated in Figure 16 below.

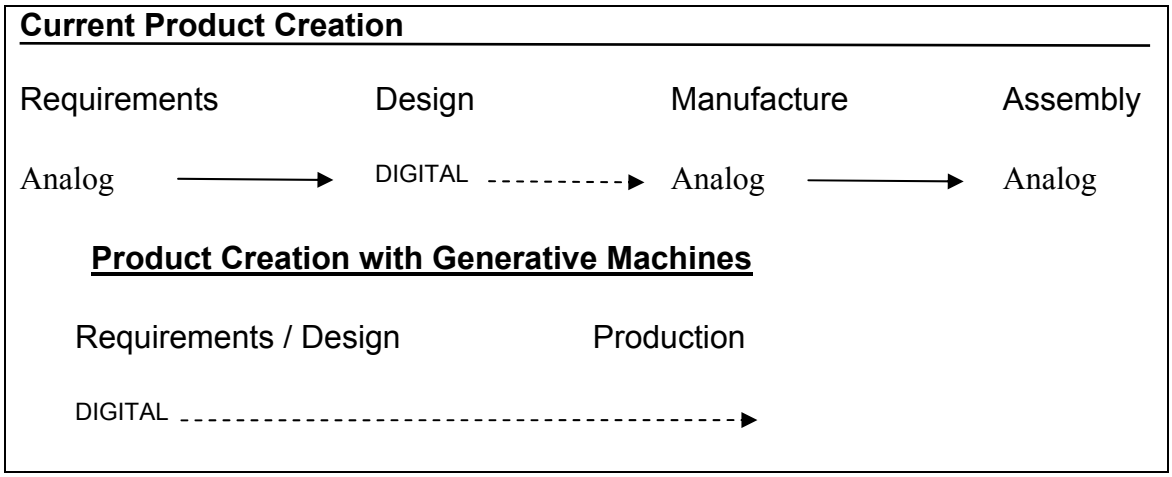


Figure 16. Fully digital product creation.

Rapid determination of the geometric limits, mechanical characteristic, manufacturing properties, etc., of physical voxels will require extensive computational exploration. In particular, the computation of shape grammar rules, which are congruent with the properties of relevant AMT, and are operated in conjunction with methods for optimization search and evaluation, could make a significant contribution to determining the potential of physical voxels. Similar computational exploration will also be required for many established materials when legal requirements for increased sustainability make it a pressing necessity to use them in conjunction with advanced manufacturing machines (i.e. not within established conventions). Also, there may be some cases in the future when a material must be used that is not known to be similar to a material that is already established in design, engineering, manufacturing and assembly. One example could be the use of indigenous materials for off-Earth building fabrication. In such cases, extensive exploration of a material's potential for enabling the fabrication of buildings with location-specific high performance geometries will be needed. Again this could be carried out through computation of shape grammar rules, which are congruent with the properties of relevant AMT, and are operated in conjunction with methods for optimization search and evaluation.

Overall, rapid exploration of materials' potential is essential to the primary sustainability goal of preserving natural ecosystems. This is because using materials within established conventions for design, engineering, manufacturing, assembly etc., will not enable radical reductions in material extraction, processing and consumption.

7.5 Rapid exploration of potential for consolidation

As discussed in section 2, AMT can be used to produce net shape consolidated assemblies at point-of-demand. This production of consolidated assemblies can take the place of several separate parts being manufactured at several different locations, and then being transported for assembly at one or more other locations. Thus, processes, as well as parts, are consolidated. In particular, the number of manufacturing, assembly, and distribution operations is radically reduced. As well as reducing the amount of materials used and energy consumed in production and distribution; the consolidation of parts and processes can reduce the complexity of product creation. Production complexity can arise from creation being both complicated and unpredictable. Product creation can be complicated because it involves many components, and unpredictable because of the vagaries of component supply. Thus, the reduction of parts and processes reduces sources of complexity in product creation. Production complexity can often lead to product defects and overtime working. This, in turn, leads to non-value adding material use and energy consumption. As discussed in section 2.3, one known example of shape grammar implementation in industry is for the routing of systems tubing through an airplane. The shape grammar was used to design several hundred tube assemblies on the 767-400ER. As discussed in section 3.4, exploration of potential for parts / process consolidation can clearly benefit from the application of generative geometric design enabled by shape grammars operated in conjunction with computational methods for optimized search and evaluation. This is because it is unlikely that the best available solution can be obtained and verified without the generation of many alterative routings. Further, there are many opportunities for part count reduction, assembly simplification and weight reduction through the application of advanced manufacturing and materials. Hence, prior knowledge of established manufacturing materials and machines is of limited usefulness. Accordingly, it would be beneficial for shape grammar rules to be formulated which are congruent with the properties of relevant AMT.

Overall, there are innumerable opportunities for meeting the primary sustainability goal of preserving natural ecosystems through consolidation of parts and processes. Moreover, many of these opportunities are particularly well suited to early adoption of shape computation. This is because they are opportunities for the consolidation of parts and processes, such as ducting, that are often a low priority for human designers.

7.6 Rapid generation of new product/component styles

As outlined in section 3, shape computation can enable unanticipated emergence of forms. Generation of unpredicted forms has the potential to provide the basis of many

new product styles that better enable people to express their potential. Also, geometric styling has the potential to increase functionality. Buildings, for example, can be better able to handle solar energy if their envelopes have unique geometries which are specific to their orientation and location. Moreover, many types of products offer improve ergonomics through better styling with unique geometries. Importantly, shape computation can enable the generation of new styles of components, such as car body panels, which must be compatible with other components, such as car engines.

7.7 Rapid creation of customer-designed branded products

As discussed in section 2, sustainable product creation should involved increased scope for self-expression among non-experts in the design and production of the physical goods that they need and/or want. However, many physical goods are valued more in acquisition, in use, and/or in resale if they have a recognized brand identity. Accordingly, many non-experts would prefer to create their own products, but have those products conform to an established brand identity. An increasing number of existing brand holders are already opening up their product development functions to ideas for new products from product users. This is being enabled by, for example, setting up web-enabled on-line communities and/or competitions. This provides brand holders with a wealth of ideas for product variations. However, it also provides them with increasing complexity in the development, production and distribution of products. An alternative would be to set up web-enabled facilities for sustainable product creation. Shape grammars, which are formulated to enable generation of original designs that conform to brand identity and regulatory requirements, could be at the centre of these web-enabled facilities. As discussed in section 3, such grammars are well-established within research. Further, these shape grammars could be congruent with the properties of those AMT which can optimize production. Furthermore, the web-enabled facilities could enable uploading of digital designs to the nearest AMT equipment. Alignment of shape computation programs with legacy CAD/CAM systems could be an important issue when, for example, the costs of transferring all of production to AMT are initially too high. In such cases, examples of alignment among computer languages during computational semantics applications can provide useful insights.

Overall, web-enabled product creation facilities driven by shape computation could enable non-experts to create products that they value themselves, and that potential future buyers would value in second hand markets. At the same time, brand holders could increase their sales and, at the same time, avoid further increasing complexity in the development, production and distribution of their products. Moreover, by using generative systems in their internal product development activities, brand holders could

accelerate the formulation of new brand identities as represented in the geometry and features of physical products.

7.8 Rapid creation of customer-designed volumetric products

Sustainable product creation involves meeting ever increasing need for volumetric social products such as low-cost housing. As described in section 3, shape grammars can be formulated that generate individual designs which are congruent with established styles of building architecture. These styles can arise from the cultural traditions of particular geographical areas and the materials which are indigenous to those areas. Accordingly, creation facilities for low-cost housing should be driven by shape grammars which are congruent with local cultural traditions. They should also be congruent with the properties of those AMT machines which can optimize production using indigenous sustainable materials. Further, shape grammars should enable the production of physical scale models which can be used by non-experts to learn how the full-sized components should be put together. Moreover, component joints should be friction-/snap-fit with parameters for tolerance, material thickness and structural modulation being included into the members of the shape vocabulary. Such forms of assembly could also be applied to the creation of other social products such as windmills.

7.9 Rapid creation of customer-designed solid products

As discussed in 2, sustainable product creation involves meeting ever increasing need for social products such as person-specific medical goods including: conformal seating; crash helmets; dental aligners; dental bridges and crowns; hearing aids; orthotic footwear; prosthetics; surgical cutting guides; and surgical implants. Compared to social products such as low-cost housing, such goods can be categorized as being solid rather than volumetric. Such goods can be much more effective if they are person-specific. However, they must conform to best practice and legal regulations. Accordingly, the time and cost of their creation can be reduced by combining shape grammars, which conform to practices and regulations, with AMT that can most efficiently produce high performance medical goods. Moreover, design data generated by shape grammar computation must be compatible with digital outputs from person-specific scans. Accordingly, examples of alignment among computer languages during computational semantics applications can provide useful insights.

A summary of opportunities for meeting primary sustainability goals through applications of AMT combined with shape computation is provided in Table 26 below. As described in section 2, sustainable means that processes are able to be carried out for

an indefinite period in such a way that individuals, organizations, societies are able to meet their needs, and express their greatest potential in the present, while preserving natural ecosystems.

Table 26. Fulfilment of sustainability goals.

Application	Principal sustainability goal
Integration of elicitation with design and production	Expressing greatest human potential
New business models	Meeting needs of people and societies
Rapid exploration of materials potential	Preserving natural ecosystems
Rapid exploration of potential for consolidation	Preserving natural ecosystems
Rapid generation of new product styles	Expressing greatest human potential
Rapid creation of customer-designed branded products	Expressing greatest human potential
Rapid creation of customer-designed volumetric products	Meeting needs of people and societies
Rapid creation of customer-designed solid products	Meeting needs of people and societies

7.10 Prelimary criteria for Generative Production Systems

Three preliminary criteria for Generative Production Systems are provided in Table 27 below. Meeting these criteria will enable realization of the potential applications outlined above. First, Generative Production Systems should encompass the elicitation of requirements, as well as design, manufacture, assembly etc. Second, they should emulate human creativity in the generation of infinite unpredicted options. Third, Generative Production Systems should exceed human capacity for generation of unpredicted options. For example, by being able to be run continuously throughout days and nights.

Table 27. Preliminary criteria for Generative Production Systems.

Criteria	Summary
Comprehensive	Enable digital elicitation, design, manufacture and assembly by non-experts
Creative	Emulate human creativity in the generation of infinite unpredicted options
Continuous	Exceed human capacity for generation of unpredicted options

8. Conclusions

The principal findings from the research reported in the preceding sections are summarized below.

- Advanced manufacturing technologies (AMT) have the potential to meet the goals of sustainable product creation: design and production of physical goods with technologies that enable individuals, organizations and societies to meet their needs and express their greatest potential while preserving natural ecosystems.
- The potential of AMT to meet the goals of sustainable product creation is currently restricted by the limitations of CAD/CAM systems which cannot enable rapid exploration of new design spaces; modeling of multi-surface, multi-material assemblies; product creation by non-experts.
- The potential of various types of generative computation to automatically produce designs has been recognized for some years. More recently, it has been proposed that generative computation can be extended from the production of designs to the production of the artifacts that are described in those designs.
- Combining AMT with generative computation to develop Generative Production Systems has the potential to overcome the current limitations imposed on AMT by typical CAD/CAM systems.
- Generative computation automatically produces options that are not stored previously in computer. These options adhere to key requirements, but are unpredictable and involve little, or no, external human input after initial programming.
- There are numerous approaches to generative computation: each with relative strengths and weaknesses. However, their relative accessibility, transferability, versatility, and functionality, make shape grammars an important source for the development of Generative Production Systems.
- Shape grammars can be considered to be a type of transformational-generative grammar which operates within production system formalisms. Through the

recursive application of transformation rules to decomposable, ambiguous, parametric, maximal, spatial elements, shape grammars can enable infinite spatial emergence.

- Shape grammars have been formulated for a wide variety of products and components, ranging from buildings to MEMS. Further, shape grammars have been formulated which relate to a wide variety of machines and materials for physical production including fuse deposition modeling and stereolithography.
- The formulation of shape grammars involves defining vocabulary, spatial relations, grammar rules, and initial shape which will generate a language of shapes. Computation of shape grammars involves definition with shape algebras, enabling with algorithms, description with pseudo-code, implementation with software.
- Challenges in the implementation of shape grammar can be grouped under the four headings of: inherent ambiguity, domain complexity, computational complexity, and software development. Similar challenges have been dealt with successfully in the computation of natural language discourses. Accordingly, the combination of computational semantics with shape grammars offers a potentially fruitful direction for the development of Generative Production Systems.
- Application opportunities for Generative Production Systems before product creation include: development of Unified Shape Production Languages which can enable the integration of elicitation with design and production; the introduction of new business models; rapid exploration of materials' potential, and parts consolidation; and rapid generation of new product/component styles.
- Application opportunities for Generative Production Systems during product creation include rapid creation of: customer-designed branded products; customer-designed volumetric products; and customer-designed solid products.
- Preliminary criteria for Generative Production Systems can be summarized as: comprehensive, creative, and continuous. In particular, enable digital elicitation, design, manufacture and assembly by non-experts; emulate human creativity in the generation of infinite unpredicted options; exceed human capacity for generation of unpredicted options.

References

Agarwal, M., and Cagan, J. (1998). A blend of different tastes: The language of coffee makers. *Environment and Planning B: Planning and Design* 25(2), pp. 205–226.

Agarwal, M., and Cagan, J. (2000). On the use of shape grammars as expert systems for geometry-based engineering design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 14(5), pp. 431–439.

Agarwal, M., Cagan, J. and Constantine, K. (1999). Influencing Generative Design Through Continuous Evaluation: Associating Costs with the CoffeeMaker Shape Grammar**,** *Artificial Intelligence for Engineering Design, Analysis and Manufacturing (Special Issue on Generative Systems in Design)*, 13(4), pp. 253–275.

Agarwal, M., Cagan, J. and Stiny, G. (2000). A micro language: generating MEMS resonators by using a coupled form – function shape grammar. *Environment and Planning B: Planning and Design,* 27(4), pp. 615–626.

Ahmad, S. and Chase, S.C. (2006). Grammar Representations to Facilitate Style Innovation: An Example From Mobile Phone Design. In V. Bourdakis and D. Charitos (eds.) *Communicating Spaces, Proceedings of the 24th Conference on Education in Computer Aided Architectural Design in Europe* (eCAADe, Volos), pp. 320–324.

Aho, A., and Ullman, J. (1972). The theory of parsing translation and compiling, Volume 1: Parsing. Englewood Cliffs, NJ: Prentice-Hall Series in Automatic Computation.

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*, pp. 521–539. Dordrecht: Springer.

Antonsson, E.K., and Cagan, J. (eds.) (2001). *Formal Engineering Design Synthesis*, Cambridge University Press, Cambridge, UK.

Beale, C.L. (2000). Nonmetro population growth rate recedes in a time of unprecedented national prosperity. Rural Conditions and Trends, 11(2), pp. 27–31.

Benros, D. and Duarte, J.P. (2009). An integrated system for providing mass customized housing. *Automation in Construction*, 18(2), pp. 310–320.

Brown, K.N., McMahon, C.A. and Sims Williams (1995). Features, aka the semantics of a formal language of manufacturing. *Research in Engineering Design*, 7(3), pp. 151–172.

Bruton, D. (1997). *A Contingent Sense of Grammar*, PhD dissertation, University of Adelaide.

Buckley, G., Henriques, M., Salazar-Xirinachs, J.M. (2008). *The promotion of sustainable enterprises*. International Labour Organization, International Labour Office, Geneva.

CAD/CAM News (2009). SIGGRAPH curated art presents generative fabrication. CAD/CAM News, 16 June.

Cagan, J. and Agogino, A.M. (1991a). Dimensional Variable Expansion – A Formal Approach to Innovative Design. *Research in Engineering Design*, 3: pp. 75–85.

Cagan, J. and Agogino, A.M. (1991b). Inducing Constraint Activity in Innovative Design. *Artificial Intelligence in Engineering Design, Analysis, and Manufacturing*, 5(1), pp. 47–61.

Cagan, J., Campbell, M.I., Finger, S., and Tomiyama, T. (2005). A framework for computational design synthesis: model and applications. *Journal of Computing and Information Science in Engineering*, 5(3), pp. 171–181.

Caldas, L. (2008). Generation of energy-efficient architecture solutions applying GENE_ARCH: an evolution-based generative design system. *Advanced Engineering Informatics*, 22(1), pp. 59–70.

Cardoso, D., and Sass, L. (2008). Generative Grammar. In J.S. Gero and A.K. Goel (eds.) *Design Computing and Cognition '08*, Springer Science + Business Media B.V, Berlin.

Ceccato, C. (2009). Liquid parametrics: fluidity of form and process. In I. Paoletti, *Proceedings of Innovative Design & Construction Technologies – Building complex shapes and Beyond (ID&CT09),* Milan, Italy, 6–7th, May, 13-25.

Chase, S.C. (1996). *Modeling Designs With Shape Algebras and Formal Logic*. Ph.D dissertation, University of California, Los Angeles, 1996.

Chase, S. C. (1997). Modeling spatial reasoning systems with shape algebras and formal logic. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 11(4), pp. 273–285.

Chase, S.C. (2002). A model for user interaction in grammar-based design systems. *Automation in Construction*, 11(2), pp. 161–172.

Chase, S.C. (2005). Generative design tools for novice designers: Issues for selection. *Automation in Construction*, 14(4), pp. 689–698.

Chase S.C. and Ahmad, S. (2005). Grammar Transformations: Using Composite Grammars to Understand Hybridity in Design, With an Example from Medieval Islamic Courtyard Buildings. In B. Martens and A. Brown (eds) *Learning From the Past – A Foundation for the Future*, Österreichischer Kunst- und Kulturverlag, Vienna, pp. 89–98.

Chau, H.H., Chen, X., McKay, and de Pennington, A. (2004). Evalation of a 3D shape grammar implementation. In J.S. Gero (ed) *Design Computing and Cognition '04*, pp. 357–376. Dordrecht: Springer.

Chen, X., McKay, A., and de Pennington, A. (2004). Packaging shape design principles to support brand identity, *Proceedings of 14th IAPRI World Conference on Packaging (WorldPak2004)*, Stockholm, Sweden, June 13–16th.

Chomsky, N. (1957). *Syntactic Structures*. Mouton: The Hague/Paris.

References

Chouchoulas, O. (2003). *Shape evolution: an algorithmic method for conceptual architectural design combining shape grammars and genetic algorithms*. Department of Architectural and Civil Engineering, University of Bath, Bath, England.

Colakoglu, B. (2005). Design by grammar: an interpretation and generation of vernacular hayat houses in contemporary context. *Environment and Planning B: Planning and Design,* 32(1), pp. 141–149.

Coyne, R.D. and Newton, S. (1989). A tutorial on neural networks and expert systems for design. In J. S. Gero and F. Sudweeks (eds) *Proceedings of Australasian Conference on Expert Systems in Engineering, Architecture and Construction*, pp. 321–337.

Datta, S. (2006). Modelling dialogue with mixed initiative in design space exploration. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 20(2), pp. 129–142.

Deak, P., Rowe, G., and Reed, C. (2006). CAD grammars: combining CAD and automated spatial design. In J.S. Gero (ed), *Design Computing and Cognition '06*, Dordrecht: Springer, pp. 521–539.

Duarte. J.P. (2005). Towards the mass customization of housing: the grammar of Siza's houses at Malagueira. *Environment and Planning B: Planning and Design, 32*(3), pp. 347–380.

Ediz, Ö. and Çağdaş, G. (2007). A Computational Architectural Design Model Based on Fractals, *Open House International* , 32(2), pp. 36–45.

Fischer, T., Burry, M., and Frazer, J. (2005). Triangulation of generative form for parametric design and rapid prototyping. *Automation in Construction*, 14(2), pp. 233–240.

Fleisher, A. (1992). Grammatical architecture. *Environment and Planning B: Planning and Design,* 8(2), pp. 221–226.

Flemming U, (1987). More than the sum of parts: the grammar of Queen Anne houses. *Environment and Planning B: Planning and Design* 14 (3), pp. 323–350.

Gero, J.S. Louis, J., and Kundu, S. (1994). Evolutionary learning of novel grammars for design improvement. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 8(2), *pp.* 83–94.

Gerzso, J.M. (2003). On the limitations of shape grammars: comments on Aaron Fleisher's article "Grammatical Architecture". In K. Klinger (ed) Proceedings of Association for Computer Aided Design in Architecture (ACADIA)*: Connecting Crossroads of Digital Discourse*, Ball State University, Muncie, Indiana, pp. 280–287.

Gips, J. (1999). *Computer Implementation of Shape Grammars*. Computer Science Department, Boston College, MA.

Gips, J. and Stiny, G. (1980). Production systems and grammars: a uniform characterization. *Environment and Planning B: Planning and Design,* 7(4), pp. 399–408.

Greiner, L. (2008). PHP, JavaScript, Ruby, Perl, Python, and Tcl Today: The state of the scripting universe. *CIO Magazine*, August 29.

Grobler, F, Aksamija, A., and Hyunjoo, K. (2008). Ontologies and shape grammars: communication between knowledge-based and generative systems. In J.S. Gero and A.K. Goel (eds.) *Design Computing and Cognition '08*, Springer Science + Business Media B.V, Berlin.

Haslinger, J., and Mäkinen, R.A.E. (2003). *Introduction to Shape Optimization: Theory, Approximation, and Computation*. Society for Industrial Mathematics, ISBN 0898715369, 9780898715361.

Heisserman, J. (1994). Generative geometric design. *IEEE Computer Graphics and Applications*, 14: pp. 37–45.

Heisserman, J.R., and Woodbury, R. (1993). Geometric designs with boundary solid grammars. In J.S. Gero and F. Sudweeks (eds) *Preprints formal design methods for CAD*, Key Centre of Design Computing, University of Sydney, Sydney, pp. 79–99.

Heisserman J and Woodbury R, (1994). "Geometric design with boundary solid grammars", in Gero J and Tyugu E (eds) Formal Design Methods for CAD (B-18)85-105.

Hiller, J., and Lipson, H. (2009). Design and analysis of digital materials for physical voxel printing. *Rapid Prototyping Journal*, 15(2), pp. 137–149.

Hornby, G.S. (2005). Functional scalability through generative representations: the evolution of table designs. *Environment and Planning B: Planning and Design,* 31(4), pp. 569–587.

Hornby, G.S. and Pollack, J.B. (2001). The advantages of generative grammatical encodings for physical design. In Proceedings of Congress on Evolutionary Computing. Seoul, Korea, May 27-30, Volume 1, pp. 600–607.

Hsiao, S. W. and Chen, C. H. (1997). A semantic and shape grammar based approach for product design. *Design Studies*,18(3), pp. 275–296.

Hsiao, S.W. and Wang, H.P. (1998). Applying the semantic transformation method to product form design. Design Studies, 19(3), pp. 309–330.

Karl, T.R., Melillo, J.M., and Peterson, T.C. (2009). Global Climate Change Impacts in the United States, Cambridge University Press, New York.

Kilian, A. (2003). Fabrication of partially double-curved surfaces out of flat sheet materials through a 3D puzzle approach. In K. Klinger (ed.) Proceedings of Association for Computer Aided Design in Architecture (ACADIA)*: Connecting Crossroads of Digital Discourse*, Ball State University, Muncie, Indiana, pp. 74–81.

Kirkpatrick, S., C. D. Gelatt Jr., M. P. Vecchi, (1983). Optimization by Simulated Annealing. *Science*, 220(4598), pp. 671–680.

References

Knight, T.W. (1992). Designing with grammars. In C. Hatch (ed.), *Computer-Aided Architectural Design*. Van Nostrand-Reinhold, New York.

Knight, T.W. (1994). *Transformations in Design: A formal approach to stylistic change and innovation in the visual arts*. Cambridge University Press: Cambridge, England.

Knight, T.W. (1996). Shape grammars; five questions. *Environment and Planning B: Planning and Design*, 26(4), pp. 477–501.

Knight, T.W. (1999). Shape Grammars in Education and Practice: History and Prospects. *The International Journal of Design Computing*, 2.

Knight, T.W. (2003a). Computing with emergence. *Environment and Planning B: Planning and Design*, 30(1), pp. 125–155.

Knight, T.W. (2003b). Computing with ambiguity. *Environment and Planning B: Planning and Design*, 30(2), pp. 165–180.

Krishnamurti, R. (1980). The arithmetic of shapes. *Environment and Planning B: Planning and Design,* 7(4), pp. 463–484.

Krishnamurti, R. (1982). *SGI: A shape grammar interpreter*. The Open University, Walton Hall, Milton Keynes, England.

Krishnamurti, R., and Earl, C.F. (1992). Shape recognition in three dimensions. *Environment and Planning B: Planning and Design,*19(5), pp. 585–603.

Krishnamurti, R., and Giraud, C. (1986). Towards and shape editor: the implementation of a shape generation system. *Environment and Planning B: Planning and Design*, 13(4), pp. 391–404.

Krishnamurti, R. and Stouffs, R. (2004). The boundary of a shape and its classification. *Journal of Design Research*, 4(1).

Krstic, D. (2001). Algebras and grammars for shapes and their boundaries. *Environment and Planning B: Planning and Design*, 28(1), pp. 151–162.

Krstic, D. (2008). Approximating shapes with topologies. In J.S. Gero and A.K. Geol (eds) *Design Computing and Cognition '08*, Dordrecht: Springer, pp. 81–100.

Larsen, G. D (2005). Horses for Courses: relating innovation diffusion concepts to the stages of the diffusion process. *Construction Management and Economics*, 23 (8) pp. 787–792.

Lee, H.C. and Tang, X.M. (2006). Generating stylistically consistent product form designs using interactive evolutionary shape grammars. *Proceedings of 7th International Conference on Computer-Aided Industrial Design and Conceptual Design, (CAIDCD '06), Hangzhou, China,* 17-19 November, pp.1–6.

Li, A. I. (2002). A prototype simulated interactive shape grammar. In *Design e-ducation: connecting the real and the virtual*, Proceedings of the 20th conference on education

in computer aided architectural design in Europe, edited by Krzysztof Koszewski and Stefan Wrona, pp. 314–317. Warsaw: eCAADe.

Lim, S., Prats, M., Chase, S., and Garner, S. (2008). Categorization of designs according to preference values for shape rules. In J.S. Gero and A.K. Geol (eds) *Design Computing and Cognition '08*, Dordrecht: Springer, pp. 41–60.

Lindenmayer, A. (1968). Mathematical models for cellular interaction in development. *Journal of Theoretical Biology*, 18: pp. 280–315.

Loukissas, Y. (2003.) *Rulebuilding: Exploring Design Worlds through End-User Programming*. MSc Thesis, Massachusetts Institute of Technology.

MacKenzie, C.A. (1989). Inferring relational shape grammars. *Environment and Planning B: Planning and Design*, 16(3), pp. 253–287.

McCormack, J.P. (2002). Designing Inner Hood Panels Through a Shape Grammar-based Framework**,** *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 16(4), pp. 273–290.

McCormack, J.P., and Cagan, J.(2002). Supporting designers' hierarchies through parametric shape recognition. *Environment and Planning B: Planning and Design*, 29(6), pp. 913–931.

McCormack, J.P. and Cagan, J. (2006). Curve-based shape matching: supporting designers' hierarchies through parametric shape recognition of arbitrary geometry. *Environment and Planning B: Planning and Design*, 33(4), pp. 523–540.

Orsborn, S. Boatwright, P., and Cagan, J. (2008). Identifying product shape relationships using principal component analysis. *Research in Engineering Design*, 18(4), pp. 163–180.

Orsborn, S., Cagan, J., and Boatwright, P. (2008a). Automating the creation of shape grammar rules. In J.S. Gero and A.K. Goel (eds.) *Design Computing and Cognition '08*, Springer Science + Business Media B.V, Berlin.

Orsborn, S., Cagan, J., and Boatwright, P. (2008b). A methodology for creating a statistically derived shape grammar composed of non-obvious shape chunks. *Research in Engineering Design*, 18(4), pp. 181–196.

Orsborn, S., Cagan, J., Pawlicki, R., and Smith, R. (2006). Creating cross-over vehicles: defining and combining vehicle classes using shape grammars. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 20(3), pp. 217–246.

Pachauri, R.K. and Reisinger, A. (2007). Climate Change 2007 Synthesis Report, Fourth Assessment Report of the Intergovernmental Panel on Climate Change. IPCC, Geneva, Switzerland.

Phillips, M.G. (2008). Framing what we see: the role of ornament in structuring Louis Sullivan's design logic. *Environment and Planning B: Planning and Design*, 35(5), pp. 772–793.

References

Piazzalunga, U., and Fitzhorn, P. (1998) Note on a three-dimensional shape grammar interpreter. *Environment and Planning B: Planning and Design*, 25(1), pp. 11–30.

Plump, D. (1999). Term graph rewriting. In H. Ehrig, G. Engles, H.J. Kreowski and G. Rozenberg (eds), *Handbook of Graph Grammars and Computing by Graph Transformation*, World Scientific, pp. 3–61.

Post, E.L. (1943). Formal reductions of the general combinatorial decision problems. *American Journal of Mathematics*, 65, pp. 197–268.

Pugliese, M. and Cagan, J. (2002) Capturing a Rebel: Modeling the Harley-Davidson Brand through a Motorcycle Shape Grammar. *Research in Engineering Design*, 13(3), pp. 139–156.

Reddy, G., and J. Cagan, (1995). Optimally Directed Truss Topology Generation Using Shape Annealing*, ASME Journal of Mechanical Design*, 117(1), pp. 206–209.

Reeves, P. (2008). Rapid (Direct Digital) Manufacturing: Moving towards a globally distributed supply chain for low volume economic production.

Rogers, E.M. (2003). Diffusion of Innovations, 5th edn, Simon & Schuster, New York.

Rudolph, S. (2006). A semantic validation scheme for graph-based engineering design grammars. In J.S. Gero (ed), *Design Computing and Cognition '06*, Dordrecht: Springer, pp. 541–560.

Sass, L. (2007a). Synthesis of design production with integrated digital fabrication. *Automation in Construction*, 16(3), pp. 298–310.

Sass, L. (2007b). A Palladian construction grammar - design reasoning with shape grammar and rapid prototyping. Environment and Planning B: Planning and Design, 34(1), pp. 87–106.f

Sass, L. (2008). A physical design grammar: a production system for layered manufacturing machines. *Automation in Construction*, 17(6), pp. 691–704.

Sass, L., and Oxman, R. (2006). Materializing design: the implications of rapid prototyping in digital design. *Design Studies*, 27(3), pp. 325–355.

Sass, L., Shea, K., Powell, M. (2005). Design Production: Constructing freeform designs with rapid prototyping. In: The International Conference on Digital Design: *The Quest for New Paradigms* (ECAADE 2005), pp. 21–24.

Schmidt, L. C. & Cagan, J. (1997). GGREADA: A graph-grammar-based machine design algorithm. Research in Engineering Design 9, pp. 195–213.

Schmitt, G., and Chen, C. C. (1991). Classes of design – classes of methods – classes of tools. *Design Studies*, 12(4), pp. 246–251.

Schmittwilken, J., Saatkamp, J., Förstner, W., Kolbe, T.H., Plümer, L. (2007). A Semantic Model of Stairs in Building Collars. *Photogrammetrie, Fernerkundung, Geoinformation PFG*, pp. 415–428.

Schön, D.A. (1987). *Educating the Reflective Practitioner*. Jossey-Bass, San Francisco, CA.

Seebohlm, T. and Wallace, W. (1998). Rule-based representation of design in architectural practice. *Automation in Construction*, 8(1), pp. 73–85.

Seo, K.W. (2007). Space puzzle in a concrete box: finding design competence that generates the modern apartment houses in Seoul. *Environment and Planning B: Planning and Design*, 34(6), pp. 1071–1084.

Shea, K., Aish, R., and Gourtovaia, M. (2005). Towards integrated performance-driven generative design tools. *Automation in Construction*, 14(2), pp. 253–264.

Shea, K., and Cagan, J. (1997). Innovative dome design: applying geodesic patterns with shape annealing. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 11(3), pp. 379–394.

Shea, K., and Cagan, J. (1999). The Design of Novel Roof Trusses with Shape Annealing: Assessing the Ability of a Computational Method in Aiding Structural Designers with Varying Design Intent. *Design Studies*, 20(1), pp. 3–23.

Shea, K., and Gourtovaia, M. (2004). *Generative designs: from algoritms to intuitive and interactive software demonstrators*. First International Conference on Design Computing and Cognition Workshop 3, Implementation Issues in Generative Design Systems, 17 July.

Shea, K., and Smith, I. (1999). Applying shape annealing to full-scale transmission tower redesign, Proceedings of the 1999 ASME Design Engineering Technical Conferences and Computers in Engineering Conference, Las Vegas, NV, DETC99/DAC-8681.

Shmueli, G., Patel, N.R., and Bruce, P.C. (2007.) *Data Mining for Business Intelligence: Concepts, Techniques, and Applications in Microsoft Office Excel with XLMiner*. Wiley-Interscience, John Wiley & Sons Inc., Hoboken, New Jersey.

Soman, A., Padhye, S., and Campbell, M.I. (2003). Toward an automated approach to the design of sheet metal components. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 17(3), pp. 187–204.

Sowa, J. (2008). Conceptual graphs. In F.v. Harmelen, V. Lifschitz, and B. Porter (eds) *The Handbook of Knowledge Representation*, Elsevier, pp. 213–237.

Speller, T.H. Jr., Whitney, D., and Crawley, E. (2007). Using shape grammar to derive cellular automata rule patterns. *Complex Systems*, 17(1), pp. 79–102.

Stiny, G. (1976). Two exercises in formal composition. *Environment and Planning B: Planning and Design*, 3(2), pp. 187–210.

References

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and Planning B: Planning and Design*, 7(3), pp. 343–351.

Stiny, G. (1991). The algebras of design. *Research in Engineering Design*, 2(3), pp. 171–181.

Stiny, G. (1992). Weights. *Environment and Planning B: Planning and Design*, 19(4), pp. 413–430.

Stiny, G. (1993). Boolean algebras for shapes and individuals. *Environment and Planning B: Planning and Design*, 20(3), pp. 359–362.

Stiny G, (1996). Useless rules. *Environment and Planning B: Planning and Design,* 23(2), pp. 235–237.

Stiny, G. (1999). Shape. *Environment and Planning B: Planning and Design*, 26(1), pp. 7–14.

Stiny, G. (2006). *Shape, taking about seeing and doing*. MIT Press, Cambridge, MA.

Stiny, G., and Gips, J. (1972). Shape grammars and the generative specification of painting and sculpture. In C.V. Freiman (ed.), *Information Processing '71*: Proceedings of International Federation for Information Processing (IFIP) Congress, North-Holland, Amsterdam, pp. 1460–1465.

Stiny, G. and Gips, J. (1980). Production systems and grammars: a uniform characterization. *Environment and Planning B: Planning and Design,* 7(3), pp. 343–351.

Stiny, G. and March, L. (1981). Design machines. *Environment and Planning B: Planning and Design,* 8(3), pp. 245–255.

Stiny, G. and Mitchell, W. J. (1978). The Palladian grammar. *Environment and Planning B: Planning and Design,* (1), pp. 5–18.

Stouffs, R. and Krishnamurti, R. (1994). The complexity of the maximal representation of shapes. In J. S. Gero and E. Tyugu, (eds) *Formal Design Methods for CAD*, Elsevier, Amsterdam, pp. 53–66.

Suppapitnarm, A., Parks, G. T., Shea, K., and Clarkson, P. J. (2004). Conceptual design of bicycle frames by multiobjective shape annealing. *Engineering Optimization*, 36(2), 165–188.

Tapia, M. (1996). From *Shape to Style, Shape Grammars: Issues in Representation and Computation*, Presentation and Selection, PhD dissertation, Department of Computer Science, University of Toronto, Toronto.

Tapia, M. (1999). A visual implementation of a shape grammar system. *Environment and Planning B: Planning and Design*, 26(1), pp. 59–73.

Ulrich, K.T. and Eppinger, S.D. (1995). *Product Design and Development*. McGraw-Hill, Singapore.

Verganti, R. (2006). Innovating Through Design. *Harvard Business Review*, 84(12), 114-122, December.

Verganti, R. (2008). Design, Meanings, and Radical Innovation: a meta-model and a research agenda. *Journal of Product Innovation Management*, 25(5), 436-456.

Wang, Y. and Duarte, J.P. (2002). Automatic generation and fabrication of designs. *Automation in Construction*, 11(3), pp. 291–302.

Wirth, N. (1976). *Algorithms + Data Structures = Programs*. Prentice-Hall.

Woodbury R. F. (1997). Shape Schema Grammars. *Working Paper*, Department of Architecture, The University of Adelaide.

Wu, Q. (2005). Bracket teaching programme: A shape grammar interpreter. *Automation in Construction*, 14(6), pp. 716–723.

Yue, K., and Krishnamurti, R. (2008). A technique for implementing a computation-friendly shape grammar interpreter. In J.S. Gero and A.K. Goel (eds.) *Design Computing and Cognition '08*, Springer Science + Business Media B.V, Berlin, pp. 61–80.

VTT Working Papers

114 Päivi Parviainen, Juha Takalo, Susanna Teppola & Maarit Tihinen. Model-Driven Development. Processes and practices. 2009. 102 p. + app. 4 p.

115 Kim Björkman, Juho Frits, Janne Valkonen, Keijo Heljanko & Ilkka Niemelä. Model-based analysis of a stepwise shutdown logic. MODSAFE 2008 Work Report. 2009. 35 p. + app. 4 p

116 Erika Holt, Hannele Kuosa, Markku Leivo & Erkki Vesikari. DuraInt-Project Workshop. Effect of interacted deterioration parameters on service life of concrete structures in cold environments. 2009 32 p

117 Paula Järvinen, Kai Puolamäki, Pekka Siltanen & Markus Ylikerälä. Visual analytics. Final report. 2009. 45 p. + app. 3 p.

118 Anna-Maija Hietajärvi, Erno Salmela, Ari Happonen & Ville Könönen. Kysyntä- ja toimitusketjun synkronointi metalli- ja konepajateollisuudessa Suomessa. Haastattelututkimus. 2009. 33 s. + liitt. 3 s.

119 Timo Korhonen & Simo Hostikka. Fire Dynamics Simulator with Evacuation: FDS+Evac. Technical Reference and User's Guide. 2009. 91 p.

120 Veikko Kekkonen & Göran Koreneff. Euroopan yhdentyvät sähkömarkkinat ja markkinahinnan muodostuminen Suomen näkökulmasta. 2009. 80 s.

121 Rinat Abdurafikov. Russian electricity market. Current state and perspectives. 2009. 77 p. + app. 10 p.

122 Bettina Lemström, Juha Kiviluoma, Hannele Holttinen & Lasse Peltonen. Impact of wind power on regional power balance and transfer. 2009. 43 p.

124 Jyrki Tervo, Antti Manninen, Risto Ilola & Hannu Hänninen. State-of-the-art of Thermoelectric Materials Processing, Properties and Applications. 2009. 29 p.

125 Salla Lind, Björn Johansson, Johan Stahre, Cecilia Berlin, Åsa Fasth, Juhani Heilala, Kaj Helin, Sauli Kiviranta, Boris Krassi, Jari Montonen, Hannele Tonteri, Saija Vatanen & Juhani Viitaniemi. SIMTER. A Joint Simulation Tool for Production Development. 2009. 49 p.

126 Mikko Metso. NoTA L_INdown Layer Implementation in FGPA Design results. 2009. 20 p.

127 Marika Lanne & Ville Ojanen. Teollisen palveluliiketoiminnan menestystekijät ja yhteistyösuhteen hallinta - Fleet asset management - hankkeen työraportti 1. 2009. 65 s. + liitt. 12 s.

129 Stephen Fox. Generative production systems for sustainable product greation. 2009. 104 p.

ISBN 978-951-38-7189-5 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 1459-7683 (URL: http://www.vtt.fi/publications/index.jsp)