The Transformation's Control and Development

C. Coppola[1], A. Calabrese[2], A. Iazzetta[3], F. Mele[2], O. Talamo[2]
[1]Department of Cultura, Facoltà di Architettura Luigi Vanvitelli, SUN, Aversa
[2]Istituto di Cibernetica "E. Caianiello", C.N.R. - Pozzuoli - Napoli
[3]Istituto Motori, C.N.R. - Napoli
e-mail:carlo.coppola@unina2.it

Abstract. *The study of DNA of artifact and the development leading to its use in the field of industrial production of a single piece is now a common feature in the syllabus of the degree in Industrial Design at Faculty of Architecture "Luigi Vanvitelli" of SUN. The Generative Design Laboratory is where this process is carried out and includes the PROGEOR project for Generative Jewels Design. The experience acquired in the Generative Design Laboratory has developed along the lines of THE SINGLE PIECE, a product which combines the uniqueness of a handcrafted artefact with mass production methods. The development of project control technologies and also production technologies enables real-life experimentation of these hypotheses to be conducted.*

Keywords. *Generative model design, ontology, agents.*

Introduction

This work was structured by investigating the order of jewellery, first seeking to establish a taxonomic classification and subsequently developing a generative procedure initially based on logic then on modeling.

The items studied were chosen in the order of jewels, more specifically a family of jewels with a circular structure, the rings (ring-like structures). These ones were the research focus for the definition of the first generative algorithm produced by the laboratory under the PROGEOR project.

The results achieved relate to the analysis and coding of possible genetic structures for families of rings, identified in current productions. This code was used to generate the evolutionary characteristics of other possible families as a test of effectiveness.

The study was conducted with particular attention to the relations between the logical-formal analysis/synthesis processes and the relation with possible developments in the implementation of Artificial Intelligence procedures and evolutionary algorithms (particularly Breeder Genetics Algorithms (DeFalco et al., 1998). This phase of research is not yet complete and is still in progress.

The formal results described appear the same, but ontologically they are very different principles. The logical generative meaning identifies two different classes of possible variations, though it was attempted to combine them. The results are self-evident at the level of the development achieved and the efforts still in progress to complete the research.

In this work an ontology is proposed as a for-

mal representation system to generate instances from corresponding classes. These classes can be described by many properties: functional, structural (spatial relations), behavioural, aesthetical, emotional ones, etc..

The jewel generation process, as many artifact generation processes, incorporate complex human activities, like the creative process. Anyway it's possible to study and manage them by means of formal methods, for instance through ontological representations (Guarino, 1998) along with suitable inference systems.

A goal of this work is to define a sufficiently general paradigm able to find some basic principles that can be captured in a computer program to help a designer (expert or not) along the design path. It should be possible to use this method to compare different approaches to the design.

The quality of the representation is here considered fundamental to let the designer acquire better consciousness of his creative artifact building activity. A frame-based ontology is used in this work as basic representation. It allows explicit and rigorous choice of the relations in the model, and enforces the designer expressivity when he adopts some (structural or functional) solution: the designer competence increases during his design activity. Furthermore, making explicit and formal the relations existing in a model reduces the number of attempts (feedback) to attain a desired artifact (see figure 1).

Ontologies as artifact generation process

An ontology-based representation can be used

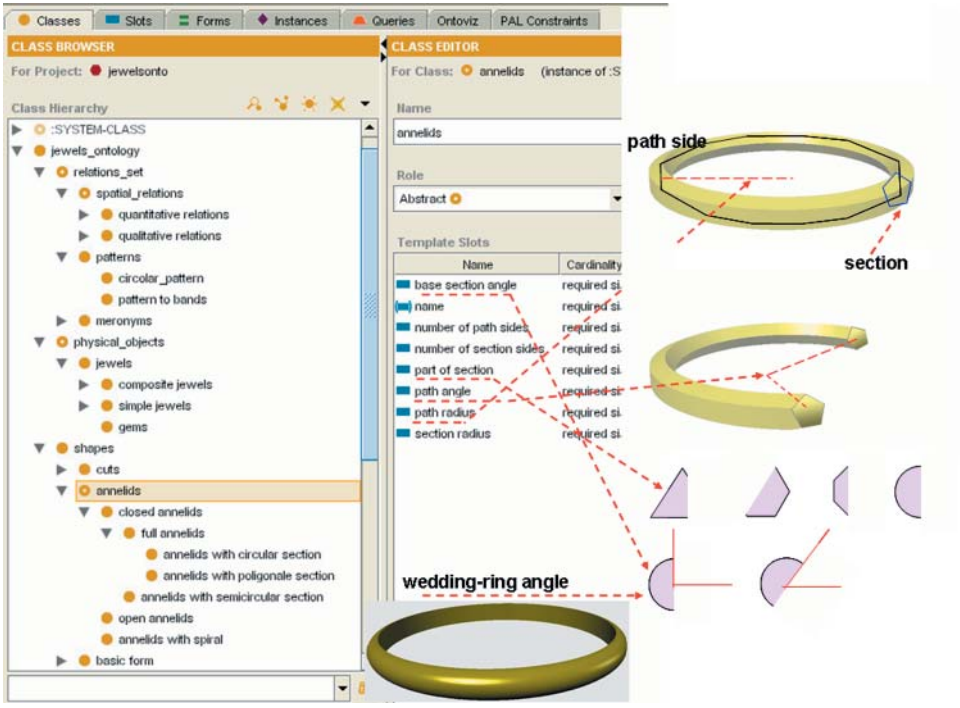


Figure 2. Definition of slots for the annelid class in Protegé

to build a formalism and consequently an axiomatics to make spatial reasoning about artifacts generation.

Simple objects

A simple object (a beam, a column, a ring, etc.) in an ontological representation is defined with a class that has a name and a set of descriptors.

The adopted formalism is based on the Frame Logic (Kifer et al., 1995) language FLORA2 (Flora2). To make easier reading the language expressions, we report its main constructs.

X::Y (X is subclass of Y), X:Y (X is instance of Y), X =>Y (X is attribute of type Y), X =>>Y (X is attribute of type Y and can have multiple values), X->Y (Y is the value of the attribute X), X->> {Y1, Y2, …,Yn } (Yi are the multiple values of X), X *=>>Y (like X =>>Y but it is not inheritable from the subclasses).

Figures 2 and 3 show the classes defined with Protegé (Protegé), a system that allows schematic and topological representation of ontologies through frames.

As an example, let us consider the following representations: the first of a beam class (parallelepiped shape, square section), described by its length and the section side, the second of a column class (cylindric shape, circular section), described by its length and the section radius.

beams[length => number_type, base_side => number_type].

columns[heigth => number_type, base_radius => number_type].

Generally speaking, a simple object class can be described as:

simple_objects[Descriptor1=>TypeDescriptor 1,Descriptor2=>TypeDescriptor2,..,].

A simple object can be a subclass of other simple objects or, directly, a subclass of the class including all the simple objects:

simple_objects::objects.

beams:: simple_objects.

columns::simple_objects.

Composite objects and ontology classes

With regard to spatial representation and reasoning, we model an artifact (a totality) through (Artale et al., 1996):

- the set of all its parts;

- the spatial (qualitative and quantitative) relations existing among its parts;

- the (possible) relations among the parts and the totality.

We represent the totality in an explicit way.

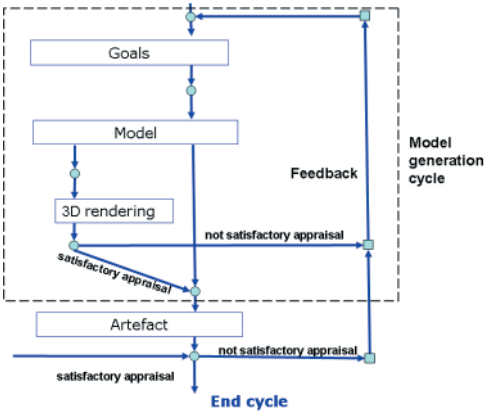
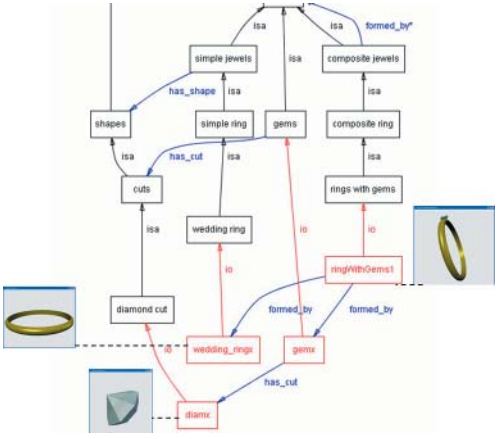


Figure 1. The cycle of artefact generation Figure 3. Topological representation of the classes through a Protegé plug-in



In other words, each totality knows its parts and the relations among them. Figure 4-a gives an example of a composite object represented by an ontology. Portals, beams and columns are classes connected by ISA relationships (class-subclass). Portalx is an instance of the portal class (IO relationship), col1 and col2 are instances of the column class and beamx is an instance of the beam class. Formed_by is a non-taxonomic relation (olonomic/meronomic).

The portal is defined by:

portalx: portals[formed_by->>{beamx,col1,col2},spatial_relation->>{r1, r2, r3, r4, r5, r6}].

with the following spatial relations:

r1:under[object1->col1,object2->beamx].
r2:under[object1->col2,object2->beamx].
r3:meets[object1->col1,object2->beamx].
r4:meets[object1->col2, object2-> beamx].
r5:dxend[object1->col2,object2-> beamx].
r6:sxend[object1>col1, object2-> beamx].

The ontology of the spatial relations

Also the spatial relations are represented in the ontology, namely every relation actually forms a class. Examples of definitions of spatial qualitative relations are (a complete sketch of the qualitative and quantitative spatial relations is given in Calabrese et al., 2004):

spatial_relation::relations.
qualitative_spatial_relations:: spatial_relations.
right:: qualitative_ spatial_relations.
left :: qualitative_ spatial_relations. over::qualitative_ spatial_relations.
under::qualitative_ spatial_relations.
behind::qualitative_ spatial_relations.
in_front_of::qualitative_ spatial_relations.

where the definition of the class qualitative_ spatial_relations is:

relations[name=> string].
qualitative_ spatial_relations [object1 => objects, object2 => objects].

Classes for the generation of artifacts

An ontology can be used as a basic representation for generating artifacts. Some examples will clarify how the generation goes on. In figure 4-b we show some variations of values of some attributes of instances of parts (beam and column) forming the portalx of the previous example.

An important set of different portals can be produced varying the spatial relations among the parts. With regard to the previous example, by canceling the relations r5 and r6 (namely the columns are not anymore constrained to be placed at the ends of the beam), we can achieve types of portal as in figure 4-c.

Obviously, without suitable constraints, objects can be generated that have no relationship with the designer class goal.

Jewel representation

Three main classes have been defined for the proposed jewels ontology:

- the class of the relations, including meronymy classes and spatial classes (namely topological, directional, geometrical, …), that can be specified among the parts composing a jewel;

- the class of the elementary shapes that define the basic parts of the jewels and basic jewels;

- the class of the physical objects to which belong the very jewels.

relations::jewels_ontology. physical_objects:: jewels_ontology. shapes::jewels_ontology.

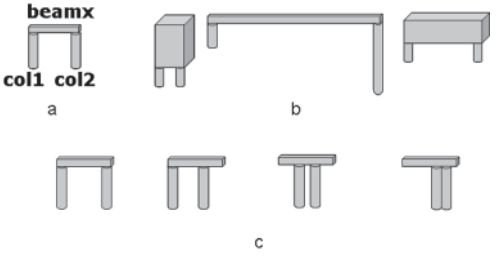


Figure 4. - a: Generation of a portal - b: Generation by variation of measures of the parts - c: Generation by variation of spatial relations among the parts

Shapes class

In the defined jewels ontology, the shape class plays fundamental role. Such a class includes some subclasses like the Gem_Cut and some classes that are primitive in the system. The last ones have a specific role to build the composite objects discussed later.

The Annelid class is described by 8 slots: name, base_section_angle, path_angle, path_sides, section_sides, section_portion, path_radius, and section_radius (in particular the slot name is inherited from the abstract class Shapes).

The slots definitions are easily derived by observing figures 2 and 3. Formally:

annelids::shapes.

shapes[name=>string].

annelids[base_section_angle=>float,number_of_path_sides=>integer, number_of_section_sides=>integer,path_of_section=>float, path_angle=>float, path_radius=>float, section_radius=>float].

In the ontology the jewel Wedding_Ring constitutes the case of circular shape annelid, with section equal to half a circle and section angle equal to 90°.

Class of simple and composite jewels

The shapes class is important in the presented ontology. The simple jewels class is connected to the shapes class by the descriptor (the slot) has_shape:

simple_jewels[name=>string, has_shape=>shapes, materials=>>material].

e.g.: wedding_ring[name -> wedding_ringx, has_shape-> annelids_to_semicircular_section, materials ->>gold].

A composite jewel in this representation is formed by parts that are simple parts, composite parts, or just other composite jewels. The definition of the composite jewel class follows:

composite_jewels[name =>string, formed_by =>> jewels, relations =>> relations_set].

Class constraints

Constraints pertaining a class (that must be true for the class and for all its derived subclasses) can be expressed in the same language of the qualitative spatial relations.

For instance, to represent a constraint pertinent to the class closed_annelid, the following Flora2 expression can be used:

constraint_violation (closed_anelids):- Cx: closed_anelids[path_angle -> X], not(X = 1.0), writeln(“constraint violation“).

(A 360° angle has been normalized as 1 and all other angles are fractions of it).

The proposed solution is simple and has the advantage of finding all the violated constraints simply by activating the following program in Flora2:

?- constraint_violation(V).

Variable V guarantees violations are searched in all the classes of the objects.

Toward a formal artifact generation model

Artifact generation is a complex activity that calls for many scientific, technical, artistic, operational skills to the designer. Therefore he must evaluate both the artifact and, reflexively, his own theoretical method and the technical process he used. We here call designer a generic human operator who drives an artifact generative process, while we will evidence what can be implemented by a computer program.

In this paragraph we report a simple artifact generation cycle, as the one for a jewel design, suitable for a formal generation model. Anyway our attention is mainly aimed to the formalization of the cycle (see figure 1).

Goals

Each generation process arises from a set of goals chosen by the designer, who is conscious of his own choices only for some goal. The artifact creator can gain (total or partial) choice consciousness during the generation process, but he can also become never conscious, not ever after he has produced and satisfactorily accepted an item.

The goals dominate the whole creative process. They can be increased, reduced or modified during the design process. They can be local - pertinent to a part of the artifact - or global - pertinent to the whole one - and of different type: structural, functional, aesthetic, emotional, etc.. The goals pertaining the structure rarely are separated from other goals pertinent to the production of an artifact. Often they are chosen to be decisive in giving a specific functional ability, an aesthetic quality, or an emotional feature to the artifact to be produced.

Also here the designer can have no design competence to foresee the effect of a particular structural choice. In other words, the designer can have no a priori knowledge if, as a consequence of the choice of a particular geometry, some goals will be respected, namely if the artifact will have the functions or desired aesthetic qualities. In the jewel generation, structural (spatial) relations and aesthetic features are of particular importance, while functional goals are less important.

These considerations are the justification for the proposal of a formal description of an artifact design cycle:

goals[agent =>name,name =>string, goal_arg=> args, gcx => float, goal_type => type].

agent Name of the designer owner of the goal (inherited slot)

name Name of the goal (inherited slot)

goal_arg Goal argument

gcx Designer goal consciousness degree

goal_type goal type (functional, aesthetic, structural, emotional)

e.g.: goals[agent -> charlesCap, goal_arg -> obx, gcx -> 0.2, goal_type -> structural].

Designer beliefs about spatial relations and goals

The designer has many beliefs during the choice of a model for the design. In the scope of this work we concentrate ourselves on the following one.

The designer believes that a chosen representation Rx can generate a (functional or structural) feature of the desired artefact (namely “it respects the goals”). We use the symbol beliefs with the same meaning as the symbol bel in Artificial Intelligence, used in the field of models of rational agents (Rao et al., 1991, Shoam, 1993).

beliefs[agent =>designerx, belief_name => name, arg => args].

e.g.: beliefs[agent -> designerx, belief_name -> nx, arg->p1].

p1: implicationProperties[headProperty-> slip_ bracelet, groupArgs ->> r1].

r1: open_annelids[path_angle-> 280].

The agent designerx believes that the goal slip_bracelet (onto wrist) can be reached by adopting the property r1 (use a path angle equal to 280° for the annelid to model).

The beliefs definition can be refined if we consider the convincement degree assigned to the beliefs itself. In such a case we must add another attribute, gcb, to the expression of beliefs.

beliefs[agent =>designerx, belief_name => name, arg => args, gcb => float].

Model

The designer uses various different models in the artifact design: a sketch on a paper, a geometrical drawing, a formal (symbolic) model, a programming language, or else a model existing only in the designer mind. Anyway the choice of a model is of crucial importance for generating the artifact. For our purposes, we will consider as a model the set of all the instances of the (simple

and composite) jewels along with the spatial relations (and with possible functional relations).

M={jx} where jx: jewels., with rk: relations_set., fs: shapes., ml: material.

(where the Frame Logic symbol ":" corresponds to the Set Theory symbol "∈").

Artifact model design states

In each step of the design cycle the set of the goals are part of the idea owned by the designer about the artifact generation state. We define this last one (named artefact_state) as the set of goals of the designer, of the structural and functional relations existing in the model, and of the beliefs of the designer.

artefact_state[agent => name, cycle => integer, groupGoals =>> goals,
groupPo =>> physical_objects, groupBeliefs =>> beliefs].

If we consider that every goal in the designer goals set is really a mental attitude (Frixione, 1994) and that the spatial and functional relations can be of type not-objective (namely they can be designer belief (in the same way of his other beliefs), then we can assert that the artifact state is a mental one. In other words the artifact model design (as to expect) is cognitive matter, therefore it can be modeled with the representation tools of the rational agents. For instance we can use the basic DBI model concepts (Desire, Belief, Intention) (Rao et al., 1991, Shoham, 1993) using the available mental constructs here available: goal (desires/aims), bel (beliefs), and int (intentions).

3D Rendering phase

In an artifact generation process the 3D rendering is not essential, while in an artifact model generation process it is: through it, it is possible to explore the global shape, the spatial relations among the parts, and it also possible to make a first evaluation of the design goal satisfaction.

Evaluation and feedback phase

It is evident from the beginning of the artifact creation process that there is a bet time, namely an abductive act, that leads to take on a new structural solution in the production cycle - a new hypothesis - that will be accepted if in produced artefact emerge the wanted structural, functional, aesthetic qualities. Hence there is a feedback that leads the designer again to reason about choices, goals, decisions already established. Then the artifact generation process has a non-deterministic evolution arising from the possible lacks of designing competences of the same designer: he could ignore that a given relation can lead to the wanted model. In the evaluation phase the designer makes a test to verify the goals satisfaction and to examine possible not foreseen new solutions. Then he starts a process of updating and revision of goals, model and beliefs. He:

- eliminates goals not anymore valid
- inserts new goals
- reviews his beliefs
- increases his knowledge degree of the goal presence

Each listed operation is carried out by comparison between the new properties observed in the 3D rendering of the object and the artifact state. The state, we remember, is just made up by designer goals and beliefs, before the rendering, that, as such, can be discussed and revised.

An example of a generation cycle

(See figure 5). The designer charlesCap starts the design cycle.

g1:goal[designer->charlesCap,goal_name ->slip_bracelet, gcx ->0.1,goal_type->functional].

m1:annelids[number_of_path_sides->52,number_of_section_sides->54, path_radius->20].

a1:artefact_state[designer -> charlesCap, step -> 1, goals -> g1, model -> m1].

The designer charlesCap observing the rendering result of the model m1 makes a revision of

his own beliefs in the following way.

b1:beliefs[designer -> charlesCap, arg->p1.].

p1: implicationProperties[headProperty->slip_ bracelet, groupArgs ->> r1].

r1: annelids[path_angle-> 280].

The designer charlesCap updates the goal consciousness and the model.

g2:goal[designer->charlesCap,goal_name->slip_bracelet,gcx->0.6,goal_type->functional].

m2:annelids[base_section_angle->30, number_of_path_sides->52,
number_of_section_sides->54, path_radius ->20, path_angle -> 280].

a2:artefact_state[designer ->charlesCap, step ->2, goals -> g2, model -> m2, beliefs ->> b1].

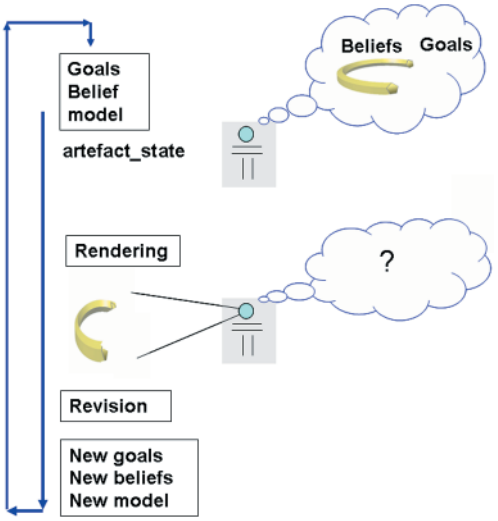


Figure 5. An example of generation cycle

References

A. Artale, E. Franconi, N. Guarino, L. Pazzi, 1996, Part-Whole Relations in Object-Centered Systems: An overview, Data & Knowledge Engineering (DKE) journal 20 347-383 – North-Holland, Elsevier, 1996

A. Calabrese, F. Mele, L. Serino, A. Sorgente, O. Talamo, 2004, Rappresentazioni di conoscenze spaziali di siti archeologici, AI*IA 2003 - Ottavo Congresso Nazionale dell'AI*IA, Polo didattico "L. Fibonacci", Università di Pisa, 23-26 Settembre 2004

I. De Falco, A. Della Cioppa, A. Iazzetta, E. Tarantino, 1998, Evolutionary Algorithms for Aerofoil Design, IJCFD, 1998, Vol. 11, pp. 51-77

Flora2 Project, http://xsb.sourceforge.net/

M. Frixione, 1994, Logica, significato e Intelligenza Artificiale, Franco Angeli, Milano, 1994

N. Guarino, 1998, Formal Ontology in Information Systems, Proceedings of FOIS'98, Trento, Italy, Amsterdam, IOS Press, 6-8 June 1998

M. Kifer, G. Lausen, J. Wu., 1995, Logical foundations of object-oriented and frame-based languages, JACM, 42(4):741--843, July, 1995.

ProtégéProject, http://stanford.protege.edu

Rao e M. Georgeff, 1991, Modelling Rational Agents within a BDI-Architecture, KR91, Ed Morgan Kaufman, San, Mateo, CA (1991).

Y. Shoham, 1993, Agent Oriented Programming, Artificial Intelligence n. 60, 51-92, Elsevier Science Publishers (1993).