INTERNATIONAL CONFERENCE ON ENGINEERING DESIGN, ICED'07
28 - 31 AUGUST 2007, CITE DES SCIENCES ET DE L'INDUSTRIE, PARIS, FRANCE

Design as Forcing: deepening the foundations of C-K theory.

Armand Hatchuel [1], Benoit Weil [2]
[1]Ecole des Mines de Paris / Fenix Chalmers, Göteborg
[2] Ecole des Mines de Paris

ABSTRACT
We present a new result about the consistency and validity of C-K design theory. C-K design theory (Hatchuel and Weil 2003) has been applied recently in a large variety of industrial contexts. Paradoxically, C-K theory offers a constructive and rigorous method in non-programmable and creative situations. To solve this paradox and warrant the properties of C-K theory we establish the correspondence between C-K theory and Forcing, a method of Set theory developed by Paul Cohen in 1963 for the "invention" of new sets. Such analysis confirms the deep relations between the two approaches. It shows that the task of C-K operators is to create potential and dynamic forcings in the real word. In return, Forcing warrants the creative process in C-K theory. It also preserves meaning when new objects are created during a design process. Taking into account the foundational aspects of Forcing in modern set theory, the correspondence between C-K theory and Forcing strengthens the foundations of C-K theory and confirms that design theory can now claim a new scientific maturity.

Key words: design theory, Set theory, design methodology, programming theory

1. Introduction: The paradoxical properties of C-K design theory.

1.1 The goals of design theory: Efforts to reach a consistent Design theory have been persistent in the last decades (Yoshikawa 1981, Reich 1995, Braha and Reich 2003). Complete automation of design is not the goal of such program (Braha and Reich 2003) and this research is not grounded on the (wrong) idea that design practice could be reduced to a pure mathematical game. Like any human collective action, Design is shaped by *managerial, social and economic forces*. However, these forces are themselves influenced by how Design is described and modelled. Therefore, progress in Design theory can at least prevent managerial, economic and social misunderstandings of Design. For instance, it is a common and illusory prejudice that all types of design should be managed according to standard Project management. This view has been critically discussed with the help of design theory (Hatchuel, Le Masson and Weil 2005). More generally, design theory has two interrelated goals. The first one is to improve the activity of designers when standard design methods are not sufficient. The second goal is to better understand the specific nature of Design when compared to classic problem solving or decision theory (Hatchuel 2003). Empirical observations tell us that Design is a dynamic process, non linear and full of surprises (Gero 1996, Braha and Reich 2003). Thus, any theoretical effort about the rationale of design has to capture these "hard" facts. The most challenging one is that design aims to generate something that is *partially unknown* and will be *progressively discovered* during the process. This is the departure point of C-K design theory.

1.2. The specific logic of C-K design theory. Recently introduced, C-K design theory (Hatchuel and Weil 2003) has attracted scholarly interest (Kazakçi and Tsoukias 2005, Salustri 2005) and has been applied in a large variety of industrial contexts.It is now documented that C-K theory *offers a constructive and practical method in non-programmable situations* i.e. in innovative contexts and early design phases (Elmquist and Segrestin 2007, Ben-Mahmoud Jouini et al. 2006, Hatchuel, Lemasson and Weil 2005, Hatchuel, Le Masson and Weil 2004). For sure, the result of innovative design cannot be predicted (or programmed) and C-K theory offers no automatic recipe. However, C-

K theory rigorously *describes* and consistently guides the operations needed to generate *new objects* presenting desired properties. Actually, C-K theory has two important benefits:
- It increases the capacity to *control* a design process, i.e. to rigorously describe each step of the process and its rationale.
- It increases the *expansion power* of a design process i.e. its capacity to be creative and to generate new solutions.
To increase simultaneously control and creativity seems paradoxical. Yet, in this paper we clarify this paradox and ground the power of C-K theory on new foundations. This needs a technical detour by modern set theory where *a powerful design theory* has been invented and can be used as a benchmark or as an extreme case for C-K theory.

1.3. Design in Set theory: the Forcing method. Attempts to model Design with mathematical tools are well documented. Yet, c*an we find design methods in mathematics? Actually,* two approaches of design exist in this field: the algebraic method and the forcing method. Forcing offers a powerful and highly general design method for Set theory. It is a universal method for designing new collection of sets which verify desired properties. The main focus of this paper is to establish *the tight correspondence* that exists between C-K theory and forcing. This correspondence offers new ground for the validation of C-K theory. Forcing also underlines the necessity to preserve meaning when new objects are defined and the need to build knowledge with *flexible and expandable definitions of objects.*

1.4. Outline of the paper. **Section 1**: We summarize the propositions of C-K theory and its main properties. We underline and illustrate the different C-K operators. **Section 2**: We briefly remind the problems of Set theory that gave birth to Forcing. Then, we introduce the main elements and properties of Forcing: the design of new sets that preserve the axioms of set theory. We interpret this result as the *preservation of meaning* in Design. **Section 3:** We establish the correspondence between C-K theory and Forcing. This warrants the consistency of C-K theory and highlights new operational conditions for C-K theory. We conclude by discussing the new status of Design theory after these findings and its implications for design practice.

2. C-K theory: results and paradoxes.

C-K theory has been introduced by Hatchuel and Weil (2003). It offers a constructive method for non-programmable design situations. In classic engineering situations, it is usually assumed that the set of admissible solutions is well defined and that design can be programmable i.e. it can be reduced to selection and validation functions. Non-programmable design situations are situations where the formulation of the design problem cannot lead to a systematic and linear sequence of algorithms that converge to a solution. Moreover, neither the set of solutions, nor the definition of a solution is given. This is a frequent situation in the development of Science based products (Hatchuel, Le Masson and Weil 2005).

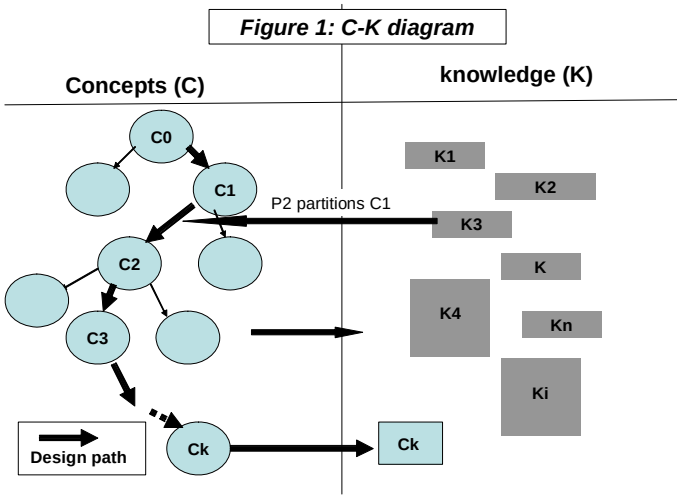
2.1. C-K theory: a brief overview of notions and operators.

2.1.1 The two spaces C and K. The name "C-K theory" mirrors the assumption that Design can be modelled as the interplay between *two interdependent spaces* having different structures and logics: the space of concepts (C) and the space of knowledge (K). The structures of these two spaces determine the core propositions of C-K theory [1]).

- **Space K** contains all established (true) propositions (the available knowledge).
- **Space C** contains "concepts" which are *undecidable* propositions in K (nor true nor false in K) about some *partially unknown* set of objects x falled a *C-set.*
- **Concepts** capture the pragmatic notion of "brief" or "broad specifications" that we find in innovative design. Therefore, concepts are propositions of the form: "There *exists some object x, for which a group of properties p1,p2, pk hold in K*".). Concepts also define *unusual sets* of objects called C-sets which elements are *not warranted in* K.

- **A design process** intends to transform such undecidable proposition into a true proposition of K. During the design process, C and K are *jointly expanded* through the action of C-K operators.

Example E: let us consider the design of "new tyres without rubber". The new concept is "*It exists a class of tyres without rubber*". Such proposition is undecidable within present standard knowledge. Existing tyres are all made with rubber (in our own knowledge); yet, there is no established truth that forbids the existence of such no-rubber tyres. Example E will be used in all sections of this paper.



2.1.2. The design process and C-K operators. According to C-K theory, design proceeds by a step by step *partitioning* of C-sets. Partitions are obtained by adding propositions (coming from K) to the concepts. Each partition adds a new attribute pi+1 to a concept Ci;, hence, Ck is the proposition "*there exists a set of objects, which verify the properties po, p1, p2… and pk*". Beginning with concept C0, This partitioning operation is repeated, whenever there is an available and compatible partitioning proposition in K and until one partitioned "C-set" becomes a "K-set" i.e. a set of objects which existence is warranted in K. With such assumptions, the following results explain the main benefits of C-K theory (Hatchuel and Weil 2003):

- Each new concept is tested in K. It is either true, false or undecidable in K. Testing a concept always expands C or expands K.
- Space C has necessarily a tree structure which follows the expansion of C0 (see Fig 1). A design solution is the concept Ck which is the first concept to become a true proposition in K. It can also be defined by the series of partitioning attributes (p1, p2 …pk) which form the *design path* that goes from the initial concept C0 to Ck. The series of sets (C0, C1, C2,..., Ck) verify ∩Ci = Ck.The other branches of C are concept expansions which do not reach a proposition that belongs to K.
- All design operations are obtained through four types of operators in each space and between spaces: namely C→C, C→K, K→K, and K→C operators. The combination of these four operators is *a unique feature of Design*. Standard models of thought and rationality are K-K operators. The four operators capture design properties including creative processes and explain seemingly "chaotic" evolutions of a real design work (Hatchuel, Le Masson and Weil 2004).

2.1.3. Expanding partitions: the generation of new objects.

A special feature of C-K theory is the existence of two types of partitions in space C: *expanding* and *restricting* ones. To distinguish between these two partitions we need to introduce *the definition of objects in K*. In example E, the attribute "made with rubber" is a common attribute of all known tyres. Therefore, the partition "without rubber" "is not a known property of the class of objects "tyres". This partition is called an *expanding partition* as it attempts to *expand* the definition of tyres by creating new tyres which are different from existing ones. In return, a partition like "a house with a red roof" is a *restricting* partition as "houses with red roofs" are well known in K. *A major advantage of C-K theory is that expanding partitions are allowed in C* whenever they form undecidable propositions. They are an essential tool for changing the definition of objects, thus the vehicle of novelty and surprise in design.

2.2. C-K theory benefits and paradoxes: the need for deeper foundations

2.2.1. C-K theory: a constructive method for non programmable problems.

In practice, the benefits of C-K theory come from its unique feature which is to offer a constructive and controllable method for non programmable problems. For sure, neither C-K theory, nor any other theory, will warrant the existence of "tyres without rubber". Yet, C-K theory describes the *necessary operations* that should be undertaken during the design process. And above all, it gives operational principles that allow *controlling the consistency* of the process.

Illustration. Example E is a design situation where no programmable method is possible and several design procedures can be suggested, for instance:

- **replace rubber** by some new specific polymer which properties seem close to rubber: In this case, design appears as *an experimental trial* (C→K);
- **Study existing tyres** and establish the functional requirements of a tyre without reference to materials: thus design appears as *a classification method* of known objects (K →K);
- **define the critical parameters** of an unknown material (K→C) which could replace rubber. Here, design appears as *the partial definition of an unknown object that could orient the search and elaboration of new material candidates* (K→K).
- **Combine several undecidable parameters** to form a new concept for instance: "a car tyre, that has not the classic shape, which uses a mix of polymers and carbon fibres, included in a metal architecture". Here, Design appears as the *creative combination of unknown objects* (C→C→C...) which could provoke the revision of the definition of tyres (K→K).

C-K theory tells us that all these propositions *are interdependent and none is consistent in itself*. To control and guide a whole design logic one has to embed these propositions in a specific Space K and elaborate a rigorous construction *of C-K expansions* of the initial concept. When such C-K modelling is done in practice the design process is actually improved.

2.2.2. Documented benefits of C-K theory in the literature.

Two major benefits of C-K modelling have been documented in real R&D situations [1]: a better control of the design rationale; and an increase of the innovative power of the design work. The second benefit usually implies the first one.

- ***Better control of the design process***: In the case of new science based products, like new unmanned aircraft vehicles, it has been shown that the design work seems to shift "randomly" its orientation. Yet, such shifts are easily understandable with C-K theory because they are the joint consequences of both concept and knowledge expansions. Thus, C-K models offer a helpful monitoring tool for such R&D projects (Hatchuel, Le Masson and Weil 2005).

- ***Increase of the innovative power of a design process***:

- During the design of an Mg-CO2 engine for Mars exploration, C-K modelling showed that Mars missions used to test the new engine concept where implicitly defined for a rover type vehicle. Then C-K modelling guided the design process towards an unexpected and innovative mars vehicle (Hatchuel, Le Masson and Weil 2004).
- Studying innovation processes in the pharmaceutical industry, it has been shown that the "drug discovery" phase, could be fruitfully modelled as a C-K design process. This approach suggested unexpected directions of investigation (Elmquist and Segrestin 2007),
- In the car industry, it has been shown that C-K theory could help to both structure an innovation policy and to foster creativity (Ben Mahmoud Jouini et al. 2006).

2.2.3. C-K theory: creative control paradox and meaning issues

- ***The paradox of creative control***: C-K theory presents a paradox: it offers, surprisingly, a constructive and practical process for non programmable design situations. It allows both controlling a design process and increasing its creative power. *What are the reasons of such paradoxical result?* To answer this question there are several possible research directions. The first one is to compare the logic and results of C-K theory with the results and of traditional creativity research and methodologies. The first results of such investigation are presented in (Lemasson, Hatchuel and Weil 2007). In this paper

we follow a different path of inquiry and we attempt to answer the following question: beyond its practical results or the likelihood of its assumptions in the field of Design, *can we warrant the validity of C-K theory by some deep property of logic?* This the core question of this paper.

- ***A hidden issue of innovative design: the preservation of meaning.*** C-K theory underlines the role of *expanding partitions* in design. Actually, these partitions raise a hidden issue. In example E, Let us assume that design succeeds. It means that a "tyre without rubber" becomes reality. Consequently, the definition of the set of objects that we call "Tyres" in K has to be changed. Assume that the first definition of a tyre was: "a special wheel for cars made with rubber". The new tyres without rubber outdate this definition and a new definition of tyres becomes necessary. Yet this new definition may impact other definitions, like the definition of wheels, and so on. More generally, C-K theory describes the generation of new objects. Yet, how can it warrant that such revisions of definitions can be done without major inconsistencies between old and new objects in K? In other words, how can we establish that the design process preserves *the meaning of both old and new things*? If these definitions are not controlled by the design process, they will become less consistent and parts of space K will loose their meaning. Finally, any design theory has not only to improve the creative power of design; it has also to preserve *meaning* i.e. the consistency of definitions in K.

In the following section we offer a novel treatment answer of these questions. We establish a direct correspondence between C-K theory and a special technique of modern set theory called "forcing". Establishing such correspondence will confirm the power of C-K theory and highlight its deep theoretical roots in common with advanced set theory.

3. Design inside Set theory: the Forcing method.

3.1. Design theory in mathematics.

Can we find design theory or methods in mathematics? If we remember that the crucial feature of design is the controlled generation of new things, two design approaches can be identified [2]. The algebraic and Forcing methods. The first one is a design technique and the second one offers a genuine design theory. The algebraic method is mostly known for the famous generation of complex numbers as an *extension* of real numbers. We know that there is no real number with a negative square. The method consists in using special properties of the division of polynomials. Let us divide any polynomial by a polynomial which has no real root (for instance $x^2 +1=0$); we thus generate new classes of equivalences built on the remainder of the division. One class of polynomials has a remainder of the form ai +b where i is the complex root of $x^2 +1=0$. These ai +b are new designed numbers which have all interesting properties of numbers and more. This is a true design technique but it is dependant of the knowledge about numbers or structures of numbers. The second approach, *Forcing*, was discovered by Paul Cohen in 1963 (Cohen, 1963 I, 1963II, 1966)[3]. It generalizes the algebraic methods[4] and is a complete theory for designing new sets that verify some desired properties. We first establish why Forcing can be seen as a design theory in Set theory; then, we establish its correspondence with C-K theory.

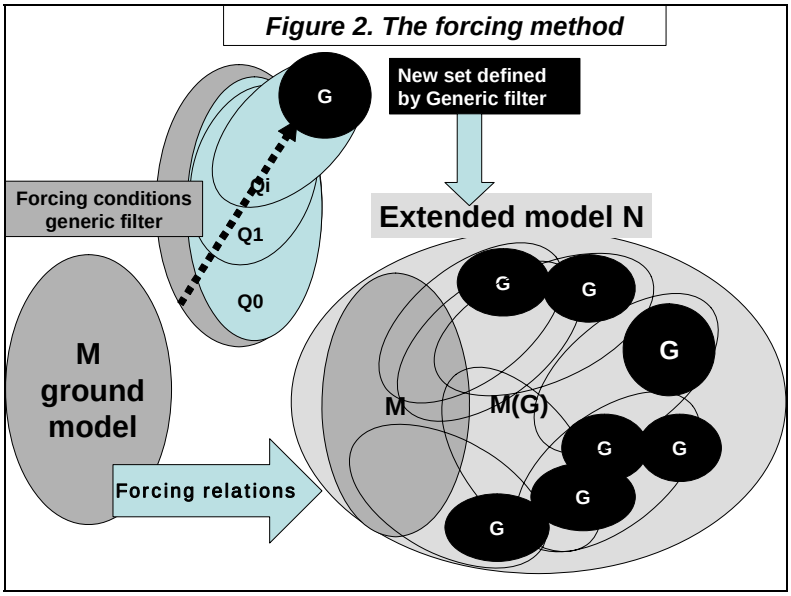
3.2. Forcing: designing new sets that preserve the definition of sets

Forcing has been described as « a *remarkably general and flexible method with strong intuitive underpinnings for extending models of set theory* » (Kanamori 1995). What means "extending models of set theory"?

- **Models of Set theory.** Set theory is built on a short list of axioms called Zermelo-Frankael axiomatic (ZF) (Jech 2003). These axioms define rules about the equality, union, separation, and well formation of sets. They also postulate the existence of some special sets. However, not all collections of sets will respect these axioms; thus, a *model of set theory* is a special collection of sets that verify ZF; it is also called *a model of ZF*. In the real world, the *definition of a thing or a class of things, (*for instance, the definition of tyres) is equivalent to a group of axioms; thus, a *model of tyres* is a collection of tyres that verify the definition of tyres.
- **Independent and undecidable propositions**. After the elaboration of ZF, set theorists faced propositions (P*) like the axiom of choice and the continuum hypothesis [5] that seemed difficult to prove within ZF. This difficulty could mean that they were *independent* from the axioms of ZF. To be

independent from ZF meant that these propositions were *undecidable* within ZF and that some models of ZF could *verify or not* these propositions. Now, proving the existence of a model of ZF that verifies a property is the same issue than proving that there is a model of tyres with no rubber. *The only possible proof in both cases is to design at least one of such models*! Actually, designing new models of ZF is not straightforward and there comes Forcing, the method invented by Paul Cohen.

3.3. The forcing method: ground models, generic filters and extensions



Forcing assumes, as a design background, the existence of a first model M of ZF, called *the ground model* and then it offers *a constructive procedure* of a new model N, called the extension model, different from M, which refutes or verifies P* and yet, is a model of ZF. In other words, Forcing is a design method that generates new collections of sets (i.e. models) and *preserves ZF.* Hence, it creates new sets but preserves meaning i.e. the well formation of all sets. A complete presentation of Forcing is beyond the scope of this paper. We must indicate that Forcing is not part of the basic knowledge for engineering Science. Therefore, we will avoid unnecessary mathematical detail and focus on the most insightful aspects of Forcing[6] which are needed to establish the central finding of this paper, *the correspondence between C-K theory and Forcing*.

331.The ground model and the forcing conditions.

- ***The first element*** of Forcing is ***a ground model M,*** a well formed collection of sets, a model of ZF. The task of forcing is to *manipulate* these sets so as to generate an extension model N which is still a model of ZF and verifies some desired property.
- ***The second element*** is the set of ***forcing conditions*** that will manipulate M. If we want to build new sets from M we have to extract elements according to some *conditions* that can be defined in M. The Forcing method builds a series of conditions on M which *step by step* completely defines **a new set which is not in M!** Let us call (Q, <) a set of candidate conditions Q and *a semi-order relation* < on Q. This set (Q;<) is completely defined in M. From Q, we can extract conditions which can form series of *compatible and increasingly refined conditions* (q0, q1, q2,...qi) where for any i : qi < qi-1; this means that each condition i *refines* its preceding one. The result of each condition is a subset of M. Hence, the series (qi) describes sets that are included in the preceding set of the series. Such series of conditions (or associated sets) is a *filter*[7] F on Q. And a filter can be interpreted as *a step by step definition of some object or some set of objects* where each step refines the preceding definition by adding new conditions.
- ***The third element of Forcing*** are the **dense subsets D** of (Q,<): *dense* subsets of Q are sets of conditions so that any condition of Q, not in D, can be refined by at least one condition belonging to

this dense subsets D. Hence, dense subsets contain conditions which combine infinitely all conditions of Q or *which cannot be refined in Q. Dense subsets* can be interpreted as sets containing *complete definitions of things (or sets) on M*.

- ***The fourth element*** of Forcing (and its essential technique) is the formation of a ***generic filter* G** (and its related set) on Q. A generic filter is a filter which *intersects all dense subsets*. If we remember that a filter is a step by step definition of a set, a generic filter warrants that this definition process reaches *the definition of a unique set or object that cannot be refined in (Q,<,M)*. Moreover as G intersects all dense subsets of Q, this new set is formed with conditions defining all other complete objects in M, provided they are compatible. Thus, *G builds a new object which is necessarily different from all objects defined in M*. Therefore, both G and this new object are not in M. We can interpret G as a collector of all information available in M and combining it to create something new not in M.
- **The *fifth element*** of Forcing is the construction method of the **extended model N**. The new set G will be used as the foundation *stone* for the generation of new sets combining systematically G with other sets of M (usually called M(G)). The union of M and M(G) is an extension model N, completely dependent on G, including M and the new collection of sets built with M. (see Fig 2).

3.3.2. The main Forcing theorems:

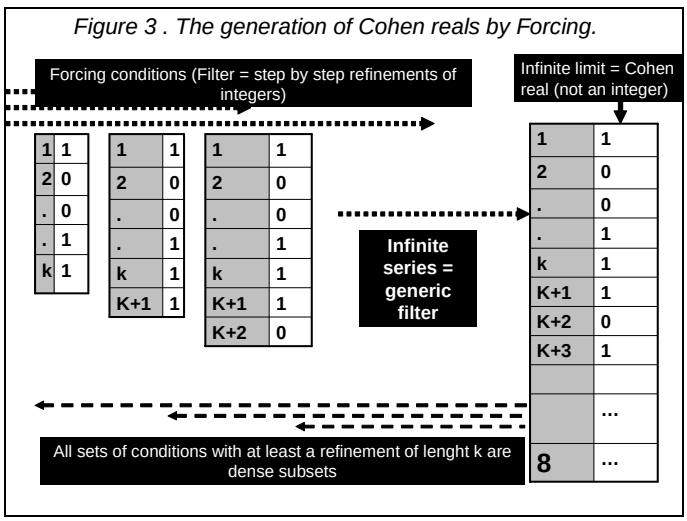
Paul Cohen invented Forcing and above all, he proved a series of remarkably general theorems that highlighted the design power of the method. The main results can be synthesized as follows:

- Forcing preserves ZF: Whenever a generic filter G exists, the new model N is a model of ZF. Hence, ZF is preserved. The new sets are not meaningless mathematical monsters.

- Forcing controls all properties of N: All properties of the elements of N are strictly dependent on the conditions (p0, .., pi) that formed the generic filter. This means that any true proposition T in N is such that there exists some pi in G so that : pi → T. Hence, the appropriate choice of the generic filter G warrants the existence of new models of sets which verify some desired properties.

An example: Forcing reals. To illustrate Forcing we give a simple application also due to Cohen (Jech 2003). It is the forcing of real *numbers* from integers (see Fig 3). *Ground model*: The set of integers ; *Forcing conditions* : Assume, a finite series of ordered integers (1,2,3,4,…, n) ; to each integer assign a random (0,1) value; we obtain a new n-list (0,1,1,1,…,0). This list can be seen as a forcing condition. The condition extracts some integers (those with value 1) and leaves the others. It also describes the set of all numbers beginning by the sequence of selected integers. Then, let us build a more refined umber definition by keeping this first list and so assigning to n+1 a (0,1) value, *without changing the values of the preceding n-list* . We obtain a new condition of length n+1 that refines the first one. The operation can be repeated infinitely. The series of conditions from length 1 to length n form a filter ; *Dense subsets*: all series of conditions that are of length >= n are dense subsets; *Generic filter*: All dense subsets contain the same limit condition which is an infinite list of selected integers. The filter which is formed with an infinite series of conditions is generic and contains this set. Hence, the generic filter builds an infinite list of selected integers that is not in M and forms a real number[8].

Figure 3 . The generation of Cohen reals by Forcing.



3.3.3. Forcing as design theory. The impact of Forcing on Set theory has been of paramount importance. « *Set theory had undergone a sea-change and beyond how the subject was enriched it is difficult to convey the strangeness of it"* (Kanamori 1996). Yet, Forcing is clearly a design theory, not an ad hoc technique. Forcing does not tell how to find (Q,<) for any properties we desire about sets. It explains rigorously *how such creative generation is possible without creating nonsense* in the world of Set theory. From our point of view Forcing is a unique case where design theory changes the realm of mathematical objects. Not only Forcing opens infinite possibilities to create new sets but it clarifies the operations that warrant *controlled novelty and meaning.* Thus, Forcing is a pure example of a constructive method for a non programmable problem; a clear case where rigour supports creativity.

This brief introduction to Forcing brings enough material to establish the deep correspondence between Forcing and C-K theory. C-K theory appears as a high level constructive protocol that generates potential forcings. In return, Forcing clarifies and confirms some important properties of C-K theory.

4. C-K theory and Forcing: a tight and enlightening correspondence.

To establish the correspondence between C-K theory and Forcing we will use a common framework that distinguishes in both cases the following items : *the design background, departure point, operators, solution, and outputs*. (Fig 4 presents a synthesis of this correspondence).

4.1. Design background: model of sets vs. knowledge space.

The design background of Forcing is a model of set theory and all semi-order relations on this model. Actually, Forcing assumes *an open amount of knowledge*. All properties that can be built on the ground model are included in this knowledge. These assumptions are equivalent to the definition of Space K in C-K theory. Space K has no ex ante boundaries. In example E, *the model of tyres* is not only the existing technical definitions of tyres *as it is the tradition in engineering departments*. Any knowledge that can be used to build attributes about tyres contributes to the model of tyres and is part of Space K. Thus forcing and C-K use very universal assumptions to define their own background. They both underline the fact that from the point of view of Design, the knowledge about an object is not limited to the definition of this object but includes all properties that can be formulated on this object.

4.2. Design departure point: set of conditions vs. initial concept.

Forcing needs a specific set of conditions (P, <) sufficiently rich and appropriate to allow building a generic filter that controls some desired property. However, Forcing is not a method to find the appropriate set of conditions. This task is left to the Set theorist [9]. Thus, *Forcing proves the existence of a constructive method for non programmable situations but it does not supply the departure point and protocol of the méthod.*. In return, C-K theory clearly begins with the formulation of a concept C0 describing some desired properties ; then, C-K operators will precisely generate a set of conditions that "force" a new set of objects which verify C0. C-K expansions will build a class of *potential* conditions (Q*, C0) which all include C0. Yet, only *a subset* of these potential conditions will finally offer a design solution.

4.3. Design solution: generic filters vs. design path.

In C-K theory, a design solution is a special path (C0,…, Ck) of the expanded tree of concepts in space C. This design path generates a series of refinements which *forms a new true proposition in K*. Whenever this series is established, several results hold.

- The partitions that form the design solution appear *compatible* in K and define a new class of object which verifies the requirement C0 (initially undecidable in K).

- Comparing with Forcing, this design path is *also a filter* as the path is generated by a step by step refinement process. Moreover, all other paths of the expansion tree being undecidable concepts, the design path is the only path of the tree that is formed by truly compatible refinements.
- The design path is also a *generic filter in C*. The set of conditions of the design solution forms a complete path in the graph from C0 to Ck (the solution concept). Along this solution path, all series of conditions that form a subpath from Ci to Ck are dense subsets of the path. Hence, the design path is a generic filter in C which includes C0 and Forces a new set of objects that verify C0.

All this propositions establish the central finding that *a design solution in C-K corresponds to a specific forcing warranting C0.*

Comment: linking genericity and engineering tolerance functions. In Forcing, the generic filter cannot be refined in (P,<) otherwise it is not generic. In C-K theory, we have established that a design solution corresponds to a generic filter. Yet, this is only true in space C, along one design path. What happens if we embed the new design solution in K? Is it still generic? There is no reason to think that the new objects formed by design cannot be refined in K only because their definition is true. So, what can be said about genericity of the design path on K? *This is where the study of Forcing provides unexpected technical validation of C-K theory.*

Figure 4. The correspondence between C-K theory and Forcing

	Forcing	**C-K theory**
Design background	M, a collection of sets, the ground model	Propositions and objects of space K
Design departure point	(Q,<) = a series of forcing conditions	The initial concept C0
Design operators	Building a generic filter G on (Q,<)	The four C-K operators and the tree-structure of concepts in C
Design solution	the new set defined by G, not in M.	a design path in C forming a new proposition in K
Design outputs	the extension model N built on G.	A new family of objects in K A set of pending concepts in C

In C-K theory, the sets in C are built in ZF without the axiom of Choice (Hatchuel and Weil 2003). In practice, like in example E, this means that design only provides the definition of a *class C of tyres without rubber*, not a complete design of a *unique tyre*. To design this unique tyre which cannot be refined in K, a choice function is needed to select it within C. Now, establishing this choice function requires *adding a new series of conditions* that will complete the design partitions generated in Space C. When this is done, we have obviously built a generic filter *on both C and K*. This seems a pure theoretical issue. Actually, we have now found a deeper interpretation of *tolerance functions* in engineering Design. These functions complement the provisional definition of an object by a condition that cannot be refined otherwise a smaller tolerance would be claimed. Thus, they warrant genericity on K and complete the forcing conditions.
This discussion also confirms the deep consistency of C-K theory, as we have found that Design has tw different tasks:
- *to leave the undecidable world of C* and warrant a design solution or path
- *to ensure the uniqueness of an object* with specific conditions [10].

4.4. Design operators: actual forcing vs. potential forcing.

Forcing is an instrument of Set theory which can be actually performed whenever generic filters exist. Instead, C-K theory describes a real world perspective. All Knowledge is not given at the beginning of the Design process and C-K operators also aim to expand this knowledge (Space K).
Example E : when design begins, available knowledge cannot warrant the existence of tyres without rubber. In forcing language, it means that there is no generic set of conditions that extends the model of tyres (tyres with no rubber). Therefore, knowledge expansion becomes a crucial tool as it *generates new potential partitions* i.e. new potential forcing conditions. For instance, introducing new materials or new knowledge about tyre shapes, cars or clients will change the dense subsets of potential partitions and prepare new potential design solutions. This expansion of knowledge organized by C-K operators may have two different impacts on potential forcings:
- *It can change the ground model* if new tyres are introduced. For instance, Tyres without air chambers or plane wheels…
- *It can change the set of conditions about tyres if some new properties of tyres are introduced*. Let us assume that we introduce in K a new material (NM) with interesting properties. The model of tyres can now be partitioned in two subsets; subset A = {tyres which are not in NM (and are in rubber)} and subset B = {tyres which are not in rubber and are in NM}.By introducing NM, we have not yet extended the model of tyres but we have changed *the way* we organize its sets and subsets***, hence the potential forcing conditions***. This operation can be repeated and the more we introduce new objects or properties, the more we increase the potential of partitions, and the more C- sets i.e.new concepts we generate until some design path is formed.

Comment: allowing expandable definitions of things in engineering design. Increasing potential Forcings is possible only if such expanding formulations are not declared false in K. Therefore, C-K theory advocates specific rules for knowledge building and for the definition of things. *Universalistic propositions should be reduced at the lowest possible level*. All propositions about things should be related to known occurrences of these things so that new partitions are not immediately rejected. This means that design requires expandable ontologies i.e. Models that can be extended by knowledge expansion (simple discovery) or by Design (Forcing). This expansion would be impossible when things are defined by abstract and universalistic definitions. This is a major issue in Engineering. Most engineering models define what is usually called conceptual models like machine elements or technical principles. Yet, they should not be used as *absolute definitions* but only as transient results of previous designs (forcings.).

Proposition : we can solidly claim that C-K theory is a protocol that dynamically *expands the forcing conditions by acting on the ground model M or the forcing conditions P.*

4.5. Design outputs. Design solutions vs. pending concepts and knowledge. At first sight, Forcing and C-K theory have the same outputs. Yet, this is not the case. Beyond design solutions and paths, the C-K process offers two other outputs : pending concepts which are incomplete sets of forcing conditions ; knowledge expansions that have contributed to these pending concepts, yet not to the present design solutions. These two types of outputs may give birth to new design solutions if Design is repeated.

5. Conclusion: a major step in the science of design.

In the preceding sections we have established that C-K design theory and Forcing present tight relations and complementary results. C-K explains how we can consistently generate new partitions which include C0, and use these partitions to expand K which again reinforces potential forcings respecting C0. Forcing ensures the generation of new objects that verify C0 and an extension of existing objects that preserves consistency and meaning. Thus, Forcing confirms the overall C-K logic and warrants its basic operational property. What are the major implications of these findings? Two of them are worth discussing in this conclusion: ***the status of design theory and some new lessons for Design practice.***

5.1. The status of Design theory.

The correspondence that we have established between C-K theory and Forcing confirms that Design theory can claim a new scientific identity grounded on the following results:

- Forcing being one of the deepest discoveries of modern set theory, we have a clear proof that design theory is a basic scientific issue.
- ***C-K theory has deep foundations*** which warrant a level of rigor and consistency rarely reached in research fields like engineering design.
- ***C-K theory is not simply applied Forcing***. It finds Forcing as a potential mechanism of a wider model of thought that describes how we rationally think when we deal with undecidable propositions and ***unknown objects***. Forcing needs to be hosted and prepared if one wants to obtain it in the real world. This is the task of C-K operators and knowledge expansion. In return, Forcing helps to clarify some neglected aspects of C-K theory like the uniqueness of the design solution and the preservation of meaning when new objects are introduced.

All these results converge to prove that Design is both a human activity and a basic class of rationality that cannot be reduced to standard learning or problem solving. This rationality is more general than other rationalities. It keeps the logic of intention but accepts the undecidability of its target; it aims exploring the unknown and it is adapted to exploit the emergent. ***The standard view that creative design cannot be organized, formalized or structured can now be strongly questioned: C-K theory, as well as Forcing, is both constructive methods in non-programmable situations.***

5.2. Implication for design practice.

This paper establishes an important theoretical result. What can we learn from it, for practice? The lesson that can be found in this theoretical detour through advanced set theory is straightforward. Now that we know that Design theory has so complex foundations, we can better understand why Design practice can be ***disconcerting, controversial and stressful***. And why such difficulties increase with the level of innovation.

Empirical research has already confirmed the complexity of Design (Blessing 2003). Yet, one cannot conclude from practice that designers lack the appropriate language and theory that allows describing it. We have to establish this language and prove that it can improve design practice. Therefore, the most practical lesson of this paper is that research has now succeeded to develop theoretical tools that can help to resist the overwhelming intellectual and social chaos that seems to emerge from design. Moreover, these same tools warrant both increased control and increased innovative power. .

Finally, Design theory has made a major step; it has achieved one of its highest scientific targets: *identify and explain a unique type of creative reasoning*. Further research should derive all fruitful theoretical and practical findings of this achievement.

References and notes.

Ben Mahmoud-Jouini S., Charue-Duboc F., Fourcade F., "Managing Creativity Process in Innovation Driven Competition," in *13th International Product Development Management Conference*, ed.

Blessing L.T., "What is Engineering Design Research?" in ICED Stockholm, Sweden: 2003).

Braha D and Reich Y., "Topologial Structures for Modelling Engineering Design Processes," *Research in Engineering Design* 14, no. 4 (2003): 185-199.

Cohen J.P. , 1963, The independance of the continuum hypothesis I., Proceedings of the national Academy of Sciences U.S.A., vol 50, 1143-1148.

Cohen J.P. , 1963, The independance of the continuum hypothesis I., Proceedings of the national Academy of Sciences U.S.A., vol 51, 105-110.

Cohen J.P., 1966, Set theory and the continuum hypothesis, Addison-Wesley.

Elmquist M., Segrestin B., "Towards a new logic for Front End Management: from drug discovery to drg design in pharmaceutical R&D," *Journal of Creativity and Innovation Management* 16, no. 2 (2007).

Gero, "Creativity, emergence and evolution in design: concepts and framework," *Knowldege-Based Systems* 9, no. 7 (1996): 435-448.

Hatchuel A., 2002, The unfinished program of Herbert Simon : towards design theory and expandable rationality, Journal of Management and Governance, 5:3-4 2002.
Hatchuel A, Weil B., (2003), "A new approach of innovative design: an introduction to C-K theory" ICED proceedings Stockholm.
Hatchuel A., Le Masson P., Weil B., (2004) "C-K theory in Practice: lessons from Industrial applications", 8th International Design Conference, Dubrovnik.
Hatchuel, A; Le Masson P; Weil B., "The Development of Science-Based Products: Managing by Design Spaces," *Creativity and Innovation Management* 14, no. 4 (2005): 345-354.
Jech T, 2003 , Set Theory, Springer monographs in Mathematics
Kazakçi O.A., Tsoukias A., (2005), "Extending C-K theory: a theoretical background for personal assistants", *Journal of Engineering Design,* 16, 4 p.399-411.
Kanamori, A., 1996, The mathematical development of set theory from Cantor to Cohen
, The Bulletin of symbolic logic pages 1 - 71.
Kunen K, 1980, Set theory. An introduction to independance proofs, Elsevier Studies in logic and the foundations of Mathematics Vol. 102 .
[19] Herbert A. Simon, *The Sciences of the Artificial*, 1991 ed. (Cambridge, MA, USA: M.I.T. Press, 1969).
Reich Y, "A Critical Review of General Design Theory," *Research in Engineering Design* 7 (1995): 1-18.
Salustri F.O., (2005), "Representing C-K theory with an action logic ", ICED proceedings, Melbourne, Australia.
Simon, H.A., 1995? "Problem forming, problem finding, and problem solving in design," in *Design and Systems: general application of methodology*, ed. A. Collen and W. W. Gasparski (New Brunswick, NJ: Transaction Publishers, 1995), 245-257.
Yoshikawa H., "General Design Theory and a CAD System," in *Man-Machine Communication in CAD/CAM, proceedings of the IFIP WG5.2-5.3 Working Conference 1980 (Tokyo)*, ed. Sata and Warman (Amsterdam, North-Holland: 1981), 35-57.

***

[1] We only refer here to applications of C-K theory that has been published in journals or refereed conferences. C-K theory is now routinely used by engineering design students in several companies (Renault, PSA, Thalés, and Decathlon). Reports of these studies are available in French, on request to the authors of this paper.
[2] The axiomatic method is an extreme form of design but it is too far from real engineering design.
[3] He has been awarded a medal Fields for this work.
[4] And the diagonal method used by Cantor to generate real numbers from integers (Cohen 1966).
[5] The two propositions of this type that gave birth to the forcing method are well known in set theory. The first one is "the existence for all sets of a choice function", also called the axiom of choice; the second one is the existence of infinite cardinals that are intermediate between the cardinal of the integers and the cardinal of the reals also called the continuum hypothesis.
[6] Complete presentations of Forcing can be easily found in standard textbooks in advanced set theory (Jech 2003, Kunen 1980, Cohen 1966)
[7] Filters are standard structures in Set theory
[8] Forcing is a mathematical tool which can design new sets using infinite series of conditions. In real design, only finite conditions are possible.
[9] For instance, to prove that the famous continuum hypothesis (CH) was independent from the axioms (ZF) of Set theory, P. Cohen "invented" a complex set of conditions that built a new model of sets where CH was false.
[10] For sure, if C0 includes the uniqueness of the design solutions the two tasks will have to be done together and genericity in C will imply genericity in K.