Identity and Discernibility of Objects: Configuration Change Management in P.L.M and Database Systems

Gilbert Giacomoni, Jean-Claude Sardas

Identity and Discernibility of Objects:

Configuration Change Management in P.L.M and Database Systems

Gilbert Giacomoni
Institut de Recherche en Gestion - Université Paris 12 (UPEC)
61 Avenue de Général de Gaulle 94010 Créteil - gilbert.giacomoni@u-pec.fr

Centre de Gestion Scientifique Chaire TMCI (FIMMM) - Mines ParisTech
60 Boulevard Saint-Michel 75006 Paris - gilbert.giacomoni@mines-paristech.fr

&

Jean-Claude Sardas
Centre de Gestion Scientifique - Mines ParisTech
60 Boulevard Saint-Michel 75006 Paris - sardas@mines-paristech.fr

Abstract: The principles and procedures for managing technical data changes specifically addressed by Product Lifecycle Management systems are examined in the context of fast changing and serially produced technical objects. Attempts to combine functionalities provided by PLM systems lead to choices that are irreconcilable with fully automated management. The expected outcomes should thus be put into perspective. The theoretical issues relate to the foundations of identity and discernibility, oneness and multiplicity, changes and invariances explored in this paper. Along the lines of E.F. Codds algebra, a formal approach to object naming is proposed in order to communicate strategies for managing relationships. The interchangeability of objects and relations, compositions and use contexts is relatively and conditionally redefined. The strategies for managing configuration changes are optimized using two generic principles that aim to preserve invariances or design new constructible ones. Those principles come into play depending on how industrial settings combine redesign (innovation) and production (serial production) rates. These investigations drew upon multiannual research conducted in manufacturing (in particular, aeronautics) as well as health care, and led to experimental corporate applications. The major PLM solutions in the market were examined.

Keywords: PLM, interchangeability, configuration, identity, discernibility, changes, invariances

Introduction

Firms must manage products, facilities and projects that are deeply complex and fast changing. This accounts for millions of definition references and even more manufactured items with successive versions and related documents throughout their lifecycles. To ensure the reliability of definitions, operations and operational maintenance, they must be able to ensure the consistency and quality of all the technical data, that is, tracking events, necessary changes, potential across-the-board impacts, non-compliance, actions, etc. These demands intensify in a diversified economy where a host of new products come and go on the market at a fast pace. To become more competitive, reduce cycle times and costs, enhance responsiveness and control increasingly complex products and processes, firms must use IT and connect, gradually or simultaneously, all the functional areas, including R&D, engineering, marketing, quality, purchases, etc. Many of them are moving into PLM-type solutions. These are enterprise tools, ERP systems both modular and integrated around a single kernel[1] (Mostefai and Batouche, 2005) that can handle product lifecycles (Batenburg and al.; 2005; Pol and al., 2005) from their design to their disposal (Stark and al. 2004; Amann, 2002). In other words, they is a modulated and integrated software set that processes and shares among the various stakeholders all the firm□s data about products and processes related to innovation and evolution. To implement these instruments and quickly reap the expected benefits, firms must make extensive efforts and investments. We felt that it was important to address these solutions to examine how the selected options for managing technical data changes operate and figure out in what ways these options meet the needs and potentially help control these changes, particularly when changes translate into many impacts on configurations. According to the promoters of PLM solutions, the key to managing technical data effectively lies in configuration management, which should provide comprehensive knowledge of the technical data related to a program, a project or a product.

[1] In an IT sense, a single kernel provides an alternative structure to heterogeneous systems interfacing software solutions having their own kernel and specific to each functional area [design, production, finance, etc.]

The following aeronautics example (source: Rome AFP) provides a snapshot of the implications. On August 6th, 2005, an ATR-72[2] airliner tragically attempted a sea landing off the Sicilian coast. Its two engines had shut down a few minutes before. The accident was mainly caused by an error in installing the fuel gauge. The ATR-72 crew thought they had 790 gallons of jet fuel. The day before a maintenance technician had to replace the gauge and the IT system suggested an ATR42 gauge, which was supposedly interchangeable in size and connection configuration. Only a small label indicated the capacity of each tank and the tank of the ATR-42 had half the capacity of the ATR-72 tank. Outdoor and in-cabin checks had detected nothing suspicious with respect to the tolerance margins of the manufacturer. So, was this computer error or human error? The interchangeability did not have an absolute but a limited value (relevant under some conditions unspecified by computers). Had we looked at the software[3] or pharmaceutical[4] industry, the issues would have been similar.

In order to manage configuration changes, PLM systems provide a wide variety of functionalities, including management by date or applicability[5] rank, updates factoring in tree structures of standardized codifications, use cases, interchangeabilities, etc. At first glance, managing the technical data of a given industry would simply require selecting the necessary and appropriate functionalities provided. This is no easy task (Eicher and al., 1984; Hatchuel, Sardas and Weil, 1988). Functionalities underpin management formalisms (for serial or unit production, with or without assembly, etc.) from

[2] EADS subsidiary, a worldwide leader in turboprop aircrafts.

[3] For example, the explosion of the first Ariane 5 rocket due (according to the report of the inquiry board) to the Inertial Reference System (IRS, which was reused from Ariane 4 (allegedly full interchangeability). Ariane 5, with its more powerful engines, tipped faster than Ariane 4 to pick up the acceleration caused by the earth's rotation. The IRS software misinterpreted this tipping of Ariane 5 as non compliant with the launch plan (of Ariane 4) and mistakenly ordered to perform a major path correction in response to a deviation that had not actually occurred.

[4] A generic medication comes with the same pharmacological substance (active ingredient), dosage form, route of administration, dosage and information (clinical research, health authorities) as the medication that it is a copy of. It is interchangeable with the original dose and therapeutically equivalent. But it is difficult to prove this and clinical studies are indirectly validated through bioequivalence. It is also the case for original medications in the event of formulation, process or manufacturing site changes.

[5] The term effectivity is also used to express the difference between applicable and applied. Effectivity conveys the quality of alignment between what one effectively does and what one intended to do.

which they originate. These functionalities spring from the software application of actual (observed) or imaginary (expected) industrial practices that served as a model for their design. Customizing the use of PLM tools designed in this way by transcribing adequately the desired activity involves choosing the right formalism(s) and modeling the activity on it/them, even if this requires adapting some practices to the formalisms or, if possible, adapting the formalisms. This touches on the subtle relationships between information systems, individuals and organizations (Marciniak and Rowe, 2008). How can a user manual of a tool adaptable to widely different manufacturing activities transcribe hybrid activities combining intensive innovation and serial production), and frequently involving many impacts on configurations?

We thus set out to understand the logic of the formalisms underpinned by PLM functionalities and how they combine for some hybrid activities, based on the premise that a firm may have to mix intensive innovation and serial production at some stage of its evolution for some or all of their business. Regarding hybrid activities, the findings suggest that the issue of managing waves of change has not been solved at a conceptual and practical level. In actual fact, attempts to combine PLM functionalities lead to choices that are irreconcilable with automated processing, and thus the expected outcomes should be considered with caution. The first section of the paper addresses this question. Accordingly, we strove to define the conditions and principles of automated management of hybrid activities, in particular using the concept of relative and conditional interchangeability. We also used a new conception of object identity and naming that binds together their composition and use configurations in an automatable way. In the second section, we explore the various theoretical and historical aspects and eventually sketch out a typology of the possible formalisms for managing configuration changes according to generic manufacturing settings variously combining innovation and serial production. This will be discussed in the second section.

1. PLM and configuration change management: industrial issues and theoretical discussion

Product Lifecycle Management (P.L.M) tools leverage cutting-edge technology that emerged in the world of aeronautics and automotive projects and spread out to other industrial sectors[6]. □PLM technology originated in the management systems of technical data (□) initially adopted in aerospace and automotive industries, and gradually spread to other more traditional industries in the early 2000s with solutions developed by companies such as Dassault Systèmes, Siemens or PTC□(Merminod, Mothe, Rowe, 2009). PLM systems offer all stakeholders a streamlined environment[7] as well as a single database for modeling[8] objects, processes[9] (Grieves 2006) and knowledge (Benbya and Meissonier 2007). They make it possible to manage items (generic term referring to a raw material, a component, a subset, a finished product), bills of materials (product tree structure), records (design, production, etc), modifications and their applicability (effectivity), configurations (variants, options, substitution, etc.), tracking down use contexts (all the products from which a part is assembled), and reuse existing data within other products, etc.

In some fields or industries[10], PLM applications have been positively evaluated both in terms of productivity and reliability of the development process. But what about firms whose existence depends on proactive innovation generating □multi-form□objects and on serial production of such objects? A □multi-form□object is a technical object that can take many forms undifferentiated by distinct names but not interchangeable. This is different from the multi-view concept that represents a technical object exploitable in various contexts[11] related to various viewpoints[12] of the business players involved in the lifecycle (Bernard, 1996). The concept of □multi-view□has generated research in PLM and product development (Gomes and Sagot, 2002; Bronsvoort and Noort, 2004; Bouiki and

---

[6] Discrete and continuous manufacturing

[7] The ergonomics of a single man-machine interface may be preferable (for trainings, application changes, etc.) to that of multiple interfaces in heterogeneous environments.

[8] Technical data, specifications, bills of materials, lines (etc.) and process sequencing.

[9] Automation of specific design or production tasks

[10] For example, home appliances within the Groupe Seb Moulinex (Merminod, Mothe, Rowe 2009)

[11] In the sense of design context [Rehman & Yan 2007] and product □configuration□

[12] Structural, technological, geometrical, functional, behavioral or contextual

al., 2008; Noel, 2006; IBM and Dassault Systèmes, 2008). The construction and evolution of each view and the control of consistency in the constructed model are still relevant issues (Bernard & Perry, 2003) as the combinatory complexity of possible relationships brings into focus relational algebra and identity semantics. The concept of "multi-form" is an object concept emerging from fast-changing environments where lead times no longer make it possible to either perform combinatory risk analysis likely to guarantee total use equivalence or to single out all scenarios as massive amounts of information build up in databases and related documents. The concept of "multi-form" enables designers to handle configurations by blending contents and contexts based on the principle of relative identity, which is somehow paradoxical from the perspective of absolute identity.

Does the user manual of PLM software accommodate usage suited to this kind of object? To answer this question, we first need to understand the various possible uses provided by PLM and how this paradoxical "multi-form" object comes into being (see 1.3).

1.1. Topping changes to make a copy compared to the original

To begin with, a technical object is simply defined as a set of interrelated elements (Simondon, 1957) that are operationally tied together. This takes the form of a list of the elements that make up the object as well as instructions on how to build it. The documentation that comes with self-assembly products is a case in point of this simple scenario. Let's assume that in the early stages the object definition (its composition and manufacturing method[13]) is shared by all stakeholders and constitutes the original and single manufacturing standard. This definition cannot remain unchanged over time because of innovation (in particular incremental innovation) and the necessity to correct errors, prevent risks and improve the quality of products. It will thus evolve as it goes through changes. These changes are processed and ranked based on generated impacts. One way to handle the successive versions of the

[13] The composition of an object restrains the connectedness of possible relationships between objects (such as "parents-children" only and supported by most information systems). But the incompleteness of the designers' algebra holds true whatever the possible initial connectedness.

definition is to measure the discrepancies with the original definition by keeping track of the changes that occurred. In this logic, the technical object is defined in its original definition along with modifications applicable to given copies. This is how the various stakeholders can coordinate their activities (Giacomoni and Sardas, 2011). Based on a date, a customer contract or a given rank (copy number of the object) usually discussed according to the existing inventory in order to limit obsolescence, any change must be recorded. The following example perfectly illuminates this practice: □Engine tuning, modification of a door or electric wire, every week (etc.), about 50 new modifications. These painstakingly rigorous operations of impact analysis on applicable documentation, use cases, outcome simulation and synthesis are performed very slowly compared to other industries. No wonder when each modification (□) translates into a change at the Kourou spaceport and specifically on the launcher. And given it is worth 130 million euros, it□s better not to go wrong. With 25 successful commercial launches for only 4 failures, in late February 2007, Ariane 5 is aiming for excellence.□ (Fodor, 2008). A simple way to understand this logic is to consider the example of a product manual intended for consumers from different countries whose respective legal provisions (norms, regulations, etc.) do not evolve harmoniously. A manual must always comply with the current legal provisions (formulation, information, disclaimers, etc.). Let□s consider that the product remains unchanged and that the manual alone must be adapted to the changing regulations in one of the covered countries. A manual designed with a language page (page 1 in French, page 2 in Russian, page 3 in Chinese, etc.) □ in the event that the applicable norm in one given country comes into effect at a given date □presents no other choice but to reprint the entire manual that must be packed into the boxes at the scheduled date, mentioning the modification n°X on page X. Accordingly, to monitor the right version of the manual corresponding to a product sold at a given date and country, it is necessary to make sure that modification n°X on page n°X is actually featured, compared to the original version. A plethora of scenarios fit into this pattern. One example is a medication (elements of a technical object) prescription (definition of a technical object) for a hospitalized patient. The prescription may change at each doctor□s visit and according to the patient□s condition. A medication can be replaced or its dosage readjusted. This is name-based management (the patient□s name), the latest version (date of last visit) of the whole prescription (for this particular patient) becomes the benchmark to avoid any clashing medication errors. This managing philosophy is essentially safety-driven and is solely appropriate for unit or short-run productions. The example of the manual, which spans the entire definition of the equipment (composition, materials, configuration, etc.) just goes to show how inconvenient and costly systematic republishing after each page modification would be. Massive documentation would be required for industrial equipment. Serial production would be impossible in these conditions. Besides, this practice poorly tolerates maintenance constraints. The replacement of a constituent part of the whole object cannot be considered separately and involves the entire duplication of the object. Replacing one page means that the whole manual should be replaced. In order to specialize productions, build responsiveness, ensure restocking and serially produce the various elements, one prerequisite is to have a definition of each element manageable independently of the object as a whole. Accordingly, the definition of each element must specify whatever change it has gone through and a set of elements must accurately build in the changes scheduled at a given rank (date, etc.). We will now discuss this issue.

1.2. Renaming the changed element in the duplicated composition: the principle of interchangeability

In contrast to the previous formalism where each object was identified as a whole, a single identification (standardized codification)[14] is now assigned to each element. It is consistent with its definition[15] and adequate to duplicate the object. If we take up the example of the multilingual manual, this comes down to designing as many different manuals as there are different languages. Each would have its own identification and be duplicated independently. Evidently, should the norm of a given country change, only the manual intended for the country involved would have to be amended and identified differently to avoid any confusion between the successive versions. But once the right copy of the manual is packed into the box, how to locate the boxes intended for the countries where the norm has changed? There are several solutions to consider. Stocks of the previous version may run out

[14] Alphanumerics

[15] Set of definition records, technical data, plans, CAD models, etc.

before stocks of the new version build up. But this solution is not always possible, for example due to restocking or maintenance. All versions must be available in the event a customer wants to replace equipment by a strictly identical model or request standard replacement. Besides, an English version is not solely limited to the country affected by a norm change. Several countries may be recipients of the English version. Reducing the quantities to be printed out by limiting the lifetime of a version only to wipe out successive versions, would be counterproductive. Another solution might be to single out the boxes based on the manual version that they contain. This solution is not always possible either. That is, if the manual accompanies equipment ultimately packed into a larger set of items, how extensive should the modification be (labeling, databases, etc.)? Besides, what if evolutions boom as a result of multiplying norms (recycling, signage, etc.) and applicable regulations? Clearly, the systematic distinction between versions can prove ineffective when the scenario becomes a little complex. A concept remedies the ripple effects of a modification, and that is interchangeability.[16] For example, the version of the manual in a box intended for English-speaking countries unaffected by normative change (for example, all but Australia) is totally indifferent. The boxes intended for these countries are interchangeable. But, in turn, it would only take a norm change in another English-speaking country (USA) for interchangeability to be reconsidered. Let's take the example[17] of a warning light, green or red, and meaning "monitoring" or "malfunctioning" depending on countries. One country can impose that all manufacturers harmonize the colors, say green, and thus the manuals must be modified accordingly. For the other countries, the signage remains indifferent. However, one of those may subsequently impose red. Clearly, the previously accepted indifference (interchangeability) is no longer appropriate.

Let's try and generalize the principles outlined in these examples. All the elements constitute a population and merely connecting them together[18] based on a given assembly logic (tree structure)

[16] NATO & ISO standards: "The ability of one product, process or service to be used in place of another to fulfil the same requirements"

[17] An example drawn from the area of life and property protection.

[18] These connections may describe an assembly breakdown, the functions fulfilled by the object within the system or, more broadly, the technical resources jointly applied and/or used for its design and production.

helps build subsets which in turn become elements and sets of subsets (now elements) corresponding to the desired technical objects. The identification of an element changes[19] along with its definition.[20] Therefore, the population of elements can be enhanced with newcomers spawned by the technological advances of their elders. A subset incorporating modifications is a new selection of modified elements. Such an element becomes, in turn, a newly identified element distinct from its contemporaries unless it is viewed as fully interchangeable with some of them, in which case it will have a common identification. In this formalism, the modified copies, no matter if they are elements or sets of elements, are differentiated by new identification; unless they are viewed as interchangeable, in which case they will have identical identification and indifferent compositions. The quest for interchangeability is one way to avoid the systematic distinction between the successive generations of elements or sets, which breaks up series and undermines the serial production strategies pursued. However, it is a radical way □with no composition record - compared to the principle of rank or date-based management that involves reviewing any change affecting an object. Can the spread of impacts of a string of modifications in the tree structures be controlled with the notion of interchangeability?

The definition of an element, subset or set includes in particular the list of its constituent elements. The definition of each constituent element provides additional information. Tracing the history of a modified element requires backtracking, tracking down in the composition the impacted constituent element that in turn passed on the impact. Thus, the composition or lineage of an element is revealed in a roundabout way. A change brings about new identification of the element and spills over into the tree structure until interchangeability is reestablished. Either a new object is defined or interchangeability obliterates any trace of composition difference at a given point. The complexity of the spreading impacts of a change through immediate neighborhood is magnified by the number of elements and connections involved and the number and frequency of the changes. In practice, it is

[19] Because identification is common to the element and its definition, any change will physically translate into a marking of the objects as well as adjustments to the related documentation shared across stakeholders.

[20] Which describes its composition and mode of production based on those elements (an element is an item of an ERP system, of a CAPM system)

impossible to survey each series change sequentially. It is a combinatory process that requires examining more definitions than there should be. The simultaneous study of a string of changes is thus necessary but it is incompatible with impact traceability. In fact, it is impossible to tolerate interchangeability for a change if it is broken by another change of the same string. And this would be irrelevant. It is thus impossible to trace an identification change when many changes have caused it.

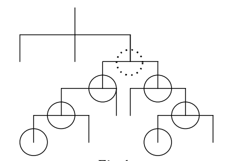


Fig.1
Parallel propagations of change impacts in a tree-structured bill of materials

Besides, the progression from neighbor to neighbor hides the crosswise impacts (on another branch of the tree structure) that can only be definitively evaluated at the common subsets level; that is, when analysis processes are already full-fledged and involve reconsidering potential interchangeabilities prematurely tolerated.

To conclude, these methods for managing definition changes are very stringent and only suited for a very moderate pace of changes, that is, in line with controlled innovation. From the perspective of serial production, it should be noted that the adoption of a formalism that assigns autonomy to the constituent elements of an object in terms of identification and evolution requires either cascading changes in order to keep track of the modified composition or interchangeability, which can be described as full since it does not keep any record of the composition changes considered as undisruptive to interchangeability. As previously mentioned, these two alternatives are incompatible with waves of change. When necessary serial production and intensive innovation combine, how to handle waves of change in the case of complex technical objects?

1.3. Relativizing interchangeabilities in the event of combinatory recomposition

The previously described formalisms relate to multi-unit activities (managing configurations and changes by rank and date including a principle of applicability) or serial activities where rhythms of innovation are monitored (updating of codified tree structures including a principle of interchangeability). In fact, the two types of activities have common configuration management that is not essentially combinatory. One reason is because the number of copies is limited, in which case the low production rate can put up with a formalism based on a monolithic definition of objects and a principle of applicability required by a fast-paced design rate. Another reason is because the number of configurations is controlled, in which case the low design rate can put up with a formalism based on a definition of objects that break down into elements and a principle of interchangeability required by a fast-paced production rate. Still, while in this case the complexity of technical objects narrowly makes it possible to study one single configuration incorporating a wave of changes - which also must be reevaluated after each wave - the formalism must accommodate these constraints with more flexibility than the two previous ones. Is this kind of formalism a combination of the previous ones or of the few other transcriptions enabled by PLM systems for these hybrid usages? Is it positively new and still to be designed?

□The parametric definition of common validity (of dependent changes toward production) is critical. It makes it easier to ground the validity of an object in the value of some of its attributes (□). The functionality of mySAP PLM is a specific implementation of a change management strategy. We are aware that this operation mode is not suitable to all businesses but it does provide a satisfactory base for most implementations. Many solutions[21] (□) boast impact analysis functionalities making it easy to track down documents, CAD data, specifications, etc. that are likely to be rethought following the modification of an object (a document, item, etc.). These functions provide major benefits. However, many solutions (□) cannot identify the business or manufacturing documents usually handled and retained in ERP systems (□). In terms of product structure management, many systems seem to provide equivalent basic functions. Still, it is important to verify if these functions are available in the delivered basic solution or if the latter is more like a toolbox, which requires the customer to build their own applications of product structure management.□ (CIM Data, 2002). Clearly, the two formalisms described up to now, applicability vs. interchangeability, cannot be reasonably linked. The applicability and interchangeability criteria have nothing in common. The former is built around a variable rank or date but including an executory principle of applicability. The latter is, by definition,

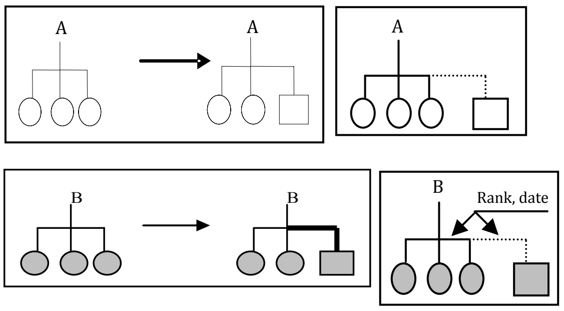
[21] In particular, those listed in the comparison table of the strategies for managing multiform configurations.

built around indifference to applicability to any specific rank or date, and thus independence from considerations of rank, date or applicability. Any change resulting in full interchangeability need not be applicable to a specific rank (on principle, this boils down to considering interchangeability as appropriate for any rank). Adopting a hybrid formalism (applicability and interchangeability) to manage a hybrid activity (serial and fast changing) would entail irreconcilable option choices that parallel information channels would necessarily support or supersede. In this scenario, the coexistence of all functional options needed by the various stakeholders (design, production, etc.) within PLM systems would imply transferring practices shared among the collective of stakeholders but contradictory all the same. And to boot, more work due to the accumulation of formalisms despite misinterpretation and error risks.

In September 2008[22], on an A330 aircraft, a defective configuration of one of the three flight control computers caused an emergency landing that required replacing the main landing gear. Airbus and the EASA[23] warned maintenance operators about the compliance with the only combinations authorized by the manufacturer: To prevent an uncertified configuration that may result in unexpected operation of the aircraft systems owners and operators should adhere to the interchangeability and mixability rules given in Airbus type certificate holder documentation In 2004, on an Airbus A340, the EASA had reported a similar problem in the airbrake operation control.

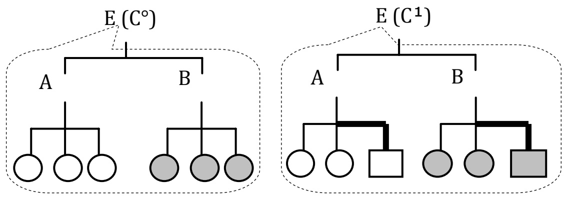
As we have seen earlier, a tree structure of elements can be defined as elements, subsets and sets (that are also elements) interrelated in a specific way.[24] When a subset A incorporates modified elements without changing identifications, it is essential to retain the match between the original tree structure and the new one (see fig-2a and 2b). There is thus a link between the subset and each original

Fig-2a – Rank or date-based BoM link management (A and B subsets)



[22] Source: Air Transport Intelligence news 11/29/08
[23] European Aviation Safety Agency
[24] This modeling is generalizable using the language of graph theory and more broadly, in set theory

Fig-2b – Use configurations of ‘multiform’ objects



element. Another link fits in between the subset and each new modified element. How to manage those links by rank and date was explained earlier. It involves activating the first links for some copies and then switching over to the new links for the following links. During the switch the first links are disabled. Date-based management operates similarly, only the switch occurs over time. These switches are scheduled to circumvent any confusion risk. But when the two links coexist, they are here defined as ‘multiform’ If the two forms are fully interchangeable, they can be chosen indiscriminately and the traceability of changes is useless. Consider a second subset B experiencing the exact same scenario as the subset A. If a subset E encompasses the two subsets A and B, there are then two scenarios:

- all combinations between the possible forms of A and B are examined to ensure full interchangeability, in which case set E including A and B will always retain the same identification E (and this will be sufficient). But, as seen earlier, this practice is essentially combinatory (number of links and elements) and thus usually unworkable.

- Only one combination is validated: A (including the new modified element) and B (including the new modified element). The identification of E is unchanged (but will no longer be sufficient). Interchangeability is only relevant between this validated configuration and the original one. The example of aeronautics provides a case in point of the consequences entailed by inadequate uses (non validated configurations).

Thus, a ‘multiform’ element is a subset or a set of elements that has the same identification even though its composition has changed. It may incorporate various generations of elements into authorized and clearly defined configurations. To manage a ‘multiform’ element, it is necessary to simultaneously handle its identification and its use configurations (that is, the initial configuration C° of set E and its following configuration C1 of the same set E). If the identification of E had changed, the issue would not come up and managing use cases would be very easy, as managing the variants and options would be. But it would be too costly to change the identification of E whenever a component (element or subset) changes. The identification of E is thus unchanged and only knowledge of C° and C1 configurations helps distinguish between the various forms of the □multiform□element. The studied and validated configurations (C° and C1) provide the authorized use configurations for the interchangeable elements and guarantee that the modified elements are present as well. As configurations change, each element is interchangeable in relation to a continuous or discontinuous sequence (C°,□ ,C1) of successive authorized configurations that are all interchangeable. This kind of interchangeability is effectively relative and *conditional*, contrasting with the total interchangeability resulting from combinatory study. Obviously, however, considering the identification of elements alone is no longer adequate to ensure the presence of the various changes of a given wave with no risk of mixing the interchangeable elements. Maintaining the match with the authorized use configurations (sequence of validated configurations) is key. Accordingly, □multiform□elements are characteristic of hybrid formalisms that cannot be prohibited or impeded. □mySAP PLM helps users build item structures (□). These structures can be interrelated to define a given product (□). The concept of validity (date, series number) helps factor in the structure changes of the product (□) [configuration control] It is possible to define product configurations and manage their validity. [the process] must be manually connected with each subset structure and each related document in the product structure. Although mySAP PLM has engineered tested and highly efficient BoM item management functionalities, these can be inadequate to address complex products with many variants. In some cases, the relations with the subsets used become confusing, which makes it harder to use the system (□)□(CIM Data, 2002). The most sophisticated PLM solutions are geared toward controlling context-driven settings (content interchangeability and use of context-driven content).[25] They make it easy to define benchmark configurations (baseline, photos, models, etc.), classes, modules or subsets of objects that are fully interchangeable [□full interchangeability□, products that are identical in all of

[25] Which results in limiting its scope of validity (restricted vs. full interchangeability)

their technically relevant properties (Form, Fit, Function [26]) or partially so (restricted interchangeability mapped between Form Fit Function classes, with special group and conditions[27] that have to be met before products can substitute for one another].

Also, the variable degree of object subdivision makes it easy to distribute structural complexity across multiple levels (subsets, modules, metadata, etc.). The strategies focus on managing the links (relationships) between these objects (modules, respectively, etc.) but they run up against the combinatory potentialities and must resort to component-reducing rules (Demoly, 2010; Boothroyd & Dewhurst, 1983). As a result, relational algebra seeks to offset the inadequacy of identity and name semantics. Occasionally, identifications [Manufacturer Part Number] may be supplemented with codes (interchangeability codes; model identification, etc.) that are common or distinctive among objects (modules, respectively, etc) according to the desired configurations. Such codes are usually extensive (depending on the subdivision degree of objects) and not physically marked. The following table compares[28] the distinctive features of the formalisms offered by the major[29] PLM systems in the discrete (vs. continuous) manufacturing market to manage multiform objects. The identity (through *full* or *restricted* interchangeability) and differentiation (through the distinctive criteria of *identification*[30] and/or *links*) of the object forms generated vary, in particular by the subdivision considered (BoMs, modules or BoM-based subsets, etc.). The search for *identities* (respectively. *differentiations*) at a given scale (micro, macro, meta, etc.) shifts the search for *differentiations* to a lower or higher level. These complex formalisms of configuration management do not operate on a

[26] Physical, functional and performance characteristics or specifications that uniquely identify a component or device and determine its interchangeability in a system (Business Dictionary).

[27] For example, in proximity to motors, only highly heat-resistant products can be used [Champion Aeropace LLC, Service Bulletin S.B. CH53536-1-74-001, Interchangeability and Intermixability of Parts, December 19th 2008]

[28] Mainly based on their functional descriptions and/or feedback (PLM Lab club, CIM Data, in particular), as well as technologyevaluation.com (comparison on 50 criteria)

[29] Particularly in terms of reputation and market shares in manufacturing: Dassault Systèmes (25%), Siemens (19%), PTC (10%) according to the US research firm Daratech in 2007.

[30] With or without extension (codes, etc.)

fully automated basis and appear to require human decision (Brown, 2006; Bouikni and al. 2008; Hwang and al; 2009).

Comparative table (1) of multi-form configurations management strategies (for discrete productions)								
	Form generation			**Processes**				
	▪ Identity of forms ▪ Differentiation of forms ▪ Automation			▪ Type of interchangeability [full/restricted] ▪ Distinctive criteria [identification / effectiveness/ modularity] ▪ Level [automatic / decision support]				
PLM solutions	**[Form identity]**	☐	**[Form differentiation]**			☐	**[Automation]**	☐
⊕ SAP (A&D)	⊕ Lufthansa Technik.; NATO HelicopterInd.; Bomber; British Airways, etc							
	Full interchangeability	☑	identification + [code][1]	☑ ☑	Effectiveness[2]	☑	Automatic	☐
	Restricted interchangeability	☑			Modularity[4]	☑	Decision support[5]	☑
Φ Lascom (ICS)	Φ Arianespace; OTAN; EADS;Thales, etc							
	Full interchangeability[6]	☑	Identification[7] + [code]	☑ ☐	Effectiveness[8]	☑	Automatic	☐
	Restricted interchangeability	☐			Modularity	☐	Decision support[9]	☑
⊕ PTC (Windchill)	⊕ GKN Aerospace Engineering Services; Oerlikon Solar; Schneider Electric; HP, etc							
	Full interchangeability	☑	identification[11] + [code]	☑ ☐	Effectiveness[12]	☑	Automatic	☐
	Restricted interchangeability	☐			Modularity[13]	☑	Decision support[14]	☑
⊕ Dassault Sys. DS Portfolio (Transcat)[19]	⊕ Boeing; Airbus; Lockheed Martin Bombardier; Pratt & Whitney Canada; Dassault Aviation, etc							
	Full interchangeability	☑	Identification[16] + [code]	☑ ☐	Effectiveness[17]	☑	Automatic	☐
	Restricted interchangeability	☐			Modularity	☐	Decision support[18]	☑
⊕ SIEMENS Portfolio[20]	⊕ Lockheed Martin Aeronautics; B/E Aerospace; Pratt & Whitney; MBDA Missile Systems; GM, etc							
	Full interchangeability	☑	Identification[21] + [code]	☑ ☑	Effectiveness[22]	☑	Automatic	☐
	Restricted interchangeability	☑			Modularity [24]	☑	Decision support[25]	☑
Excerpts of functional descriptions in appendix 1								

How deeply can these solutions be automated? The most glaring obstacle is to single out the various forms of the physical object whose identification remains unchanged. For one given operator, identification alone may reveal or conceal the object's composition. And this composition (subset) is vulnerable to assembly configurations (over-set). All PLM potentialities cannot be leveraged until this obstacle is eliminated (Merminod, 2007). It will still be impossible to automatically compute the collection and planning of needs or to automatically search for all the serialized elements that must be updated (retrofits), and obtain the full benefits of economies of scale. The coexistence of various generations of technical objects mixed together in databases, workshops or warehouse facilities will require differentiating between them (the modified elements) and grouping them together (the invariant elements). PLM systems will only provide stakeholders with incomplete or undecidable solutions, helping with analysis and decision-making, requiring costly labor assigned with monitoring the differences between planning and actual operations to be carried out, between available elements and expected identifications as well as their location, or even with checking for configuration gaps. While inspections will not be necessarily detrimental to the quality of equipment, they will result in similar proportions to error reports, deviations and even, occasionally, tricky element substitutions. It will always be difficult to initiate an upgrading of the whole range of equipment whatever its successive versions, locating the elements and their movement across the facilities and warehouses, or managing the availability of elements. A global and common picture of the information enabled by a consistent and dynamic tool is ultimately a necessary (how else to manage relationships between fast changing objects throughout their lifecycles?) but insufficient prerequisite. The equivalence relation between subsets and over-sets should translate into a redefinition of object identity (and therefore identification). This issue goes beyond the PLM population as it relates to the fundamentals of database management (Codd, 1970, 1990), which deserve attention here.

1.4 Theoretical issues common to database management: foundations of sets identity and discernibility

All the formalisms of configuration management common to PLM systems and, more broadly, in database management systems[31] or even software engineering tools[32], seek to manage dynamically the links (relations, interfaces, edges, etc.) between the elements (modules, subsets, etc.) resulting from the breakdown of objects (applications, systems, etc.) and their successive generations (variants, options, models, classes, etc.): *source control*[33], *generation tools*[34], *continuous integration*[35], *integrated environment*[36], etc. The implications are similar and the literature abundant (Raymond E.S.,

[31] Information stored in a computer system and organized to be consulted, edited, duplicated, saved or even restored according to a usually relational model. A database management system (DBMS) is software that permits these operations. Typically, it is simultaneously used by other software as well as administrators or developers.

[32] Product design and implementation activities and procedures which tend to rationalize the production and tracking of software [J.O ; 02/19/84]. The website LWN.net published analysis of the contributions to the Linux kernel over a year (2.6.16 through 2.6.20) : 28,000 changes added by 1,961 different developers, replacing 1,660,000 lines by 2,010,000 lines of new code ; the kernel built in 754,000 lines.

[33] Adding simultaneously hundreds of developers by nailing down each change, its author, date and purpose.

[34] To automate program generation operations by managing dependencies between components.

[35] To generate and control the entire application, in order to identify as much in advance as possible the potential regressions, errors or module incompatibilities (resulting from the partitioning of the system into subsystems, classes, objects and functions)

[36] To accommodate a host of extensions.

1998; Bellagio and Milligan, 2005; Stark, 2004; Djezzar, 2003; Neagu and Faltings, 2001; Sacquet and Nowencien, 1995; Ghoul, 1983). The National Institute of Standards and Technology (NIST) estimated at $60 billion the losses incurred by US manufacturing and commerce due to bugs held in software (S&T Press USA-n°324 □sept.2002). □The more complex the digital resource, the greater the potential loss is likely to be. For example, interchanging the data held in geographical information system (GIS) databases and groupware databases could involve the loss of thousands of links that have taken years of effort to create and which represent the bulk of the value of the database□ (Feeney, 1999).

In a seminal article from 1970 on relational databases, E.F. Codd addresses the issue of configuration management as follows: □□Many of the existing formatted data systems provide users with tree-structured files or slightly more general network models of the data. Application programs developed to work with these systems tend to be logically impaired if the trees or networks are changed in structure (□) Systems which provide users with a network model of the data run into similar difficulties (□) Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed.□ (Codd, 1970). Interchangeability and □multiform□ objects are central to this issue: □(□) The totality of data in a data bank may be viewed as a collection of time-varying relations (□). Accordingly we propose that users deal, not with relations which are domain-ordered, but with *relationships* which are their domain-unordered counterparts. In mathematical terms, a relationship is an equivalence class of those relations that are equivalent under permutation of domains□ (Codd, 1970). Note that a domain [d] is a countable set of values characterized by a name and that a relation [R] is a subset of the Cartesian product[37] of a list of domains characterized by a name. Concatenating (without information loss) two relations (R and S) that have at least one common domain can reveal an element (□1□) of the common domain □(□) which gives rise to the plurality of

[37] A Cartesian product of a domain list [D1, D2,□ ,Dn] is the set of distributions of possible values (n-tuples) respectively selected one by one in each domain [(v1, v2, vn) ; (wl,w2, wn) ; etc.]

Table 2 Extraction from the example of E.F. (Codd, 1970)

Domain 1 (supplier)	(Common) Domain 2 (part)	Domain 3 (project)
Relation 1 (R)	Relation 1 (R) / Relation 2 (S)	Relation 2 (S)
1	1	2
2	1	1
2	2	1

joins. Such an element in the joining domain is called a point of ambiguity ( ). A function is a binary relation, which is one-one or many-one, not one-many(table 2).

Table 3 Cartesian Product and One-Manytype relation

Domain 1 - ε = {A,B}	Domain 2 - ε* = {1,2,3}	ε x ε*	
A	1	A	1
B	2	A	2
Primary key	3	B	1
		B	2
		C	1
		C	2
		Foreign key	

A relation is constructed from a Cartesian product that may generate a one-manytype relation, as the example below shows (table 3). In fact, when it comes to identifying each line of a table in one way (related elements), a key [Primary Key] is required. Typically, it is an additional column[38] assigning distinctive identification numbers, with no replication or blank space [subrogate primary key vs. Foreign key]. Objects identified in one way: Both programming and non-programming users perceive all objects to be identified in exactly one way, whether these objects are abstract or concrete and whether they are so-called entities or relationships(Codd, 1990).

Oneness and multiplicity, discernibility, naming and equality (interchangeability) of named things are critical issues for E.F. Codds algebra as well as set theory (Russel, 1903; Jech, 1978) on which this algebra is based: One important effect that the view adopted toward data has on the language used to retrieve it is in the naming of data elements and sets ( ) The adoption of a relational model of data ( ) permits the development of a universal data sublanguage based on an applied predicate calculus[39] ( ) Predicate logic took 2,000 years to develop, beginning with the ancient Greeks who discovered that the subject of logic could be intelligently discussed separately from the subject to which it might be applied, a major step in applying levels of abstraction(Codd, 1970). He also proposes a form of generic identification but this avenue of reflection remains incomplete (we pick it up in the second part). The simple form[40] *R.d* will often be adequate ( ) In the remainder of this paper, we shall not

[38] Identitycolumnfor SQL/Server, autonumberfor Microsoft Access, sequencefor Oracle, etc.

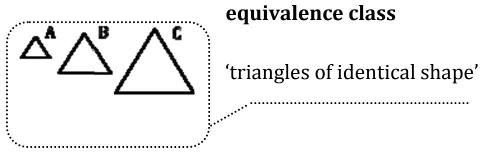
[39] Defining which are the valid statements (via symbols, *variables*, *relations*, logical connectives, etc.) and which are not.

[40] *R* is a relation and *d* a domain.

bother to distinguish between relations and relationships (□) while it is not the purpose of this paper□ (Codd, 1970).

Set theory defines the primary form of the identity of objects: □I use the term □set□or □system□to generally mean any multiplicity that may be conceived of as a unity, that is, any collection of determined elements that, through a law, may be combined into a whole□(Cantor, 1883). Identity is designed by switching from the □collection□[41] to the set, from the multiple [□many□: foreign key] to the unique [□one□; primary key]. The conception of equality (e.g interchangeability) requires switching from one environment back to the other. That is, in a set environment, two identical things are indiscernible: □By set, we mean any collection of definite, distinguishable objects of our intuition or of our intellect to be conceived as a whole. The objects are called the elements or the members of S□ (Cantor 1895). □A set can be described metaphorically as a □primary□box containing □secondary□ boxes that never have equivalent contents, elements that in turn contain tertiary boxes themselves containing and so forth□(Godement 2001). The naming of equal things ties in to the paradox of their discernibility: □Until predicates have been assigned, the two substances remain indiscernible; but they cannot have predicates by which they cease to be indiscernible, unless they are first distinguished as numerically different□(Russel, 1900). To conceive equality is to conceive a new equivalence class[42] that is not a priori named: □□from similarity between elements it is possible to derive [by abstraction] another concept to which no name has yet been given. Instead of □the triangles are similar□we say that □the two triangles are of identical shape□ or □the shape of one is identical with that of the other□ (Frege, 1893). This □abstraction□derived conception sits at the juncture of the collection (classes) and the set: □The prevalent view is that abstracts should just be treated as equivalence classes (...) The theory of abstraction thereby becomes a part of the much more comprehensive theory of sets or classes□(Fine, 2002).

Fig 3 - conception of an equivalence class



[41] The term collection is typically used to mean a set where order is ignored but multiplicity matters.

[42] The notion of class generalizes that of set. In the case of objects described in the form of relations between elements (graph), an equivalence class is a set.

Going back to the origins of object identity makes it possible to reconsider the conceptual and practical difficulties of configuration management when duplication and transformation rates accelerate. The strategies for naming objects and equivalence classes may offer an alternative (or complement) to relation management strategies.

2. Conceiving relative identity and management strategies

Relation management does not align easily with the issue of the interchangeability of two sets (representing objects, etc.), that is, their identity, because it conveys a quest for independence from their respective contexts. In fact, the equality of two sets (that is, their identity) is established by mutual inclusion as each set provides a context for the other: “If A is included in B and B is included in A, then A is equal to B and vice versa” The quest for independence, by definition, thus requires an ability to dispense with any relation of contextual dependence. In this respect, it would be paradoxical to look through the relation management strategies alone for a way to automate the processing of interchangeabilities and configurations. Accordingly, we propose to explore alternative strategies for name management and lay down the prerequisites that we feel are necessary and adequate to enable automated processing within PLM systems. In fact, one of the purposes of PLM is to offer mechanisms, particularly for identifying and modeling information related to products and processes (Abramovici, 2007; Reix and Al, 2011). First, we propose to discuss the issue of identity and discernibility by differentiating between these two concepts that are fused in set theory. Then we tackle the issue of interchangeability through its ability to absorb configuration developments by redesigning invariances hard-wired into their identity and discernibility. This issue is exemplified by historic milestones such as the advent of printing (standardization through a molding method for duplicating multiple and interchangeable types) and horology (conception of the matrix to absorb the effects of innovation). Finally, we outline the principles of a typology of correspondences between some industrial dynamics (detectable from redesign and duplication rates) and the strategies that are likely to suit them in order to manage configuration developments (relations and identity management).

2.1 Sets identity and discernibility, a conception of invariance

The strategies for managing configuration changes focus on the relations between elements, subsets and sets[43] as well as on their assigned names. The identification and composition laws[44] of the entities that make up the desired objects spring from the choice strategies[45] actionable on link and name populations that change more or less rapidly. In the field of chemistry, biology or genetics, those laws can differ substantially from the field of software or aeronautics as the norms, interpretations and usages are fairly different. However, it is all about working out an algebra and semantics that enable designers[46] to describe the successive states of the definition of objects □whose generations coexist □ including their changes and invariances (Mantripragada and Whitney, 1999). Obviously, the timescale interacts with the capacity to handle representations that are structurally relevant, identifiable, recognizable and duplicable for the various stakeholders. According to Aristotle[47], □the philosophy of nature□is the □study of changing things□(Kosman, 1969).

The primary issue is that of the identity of an object and the equality of two objects (identical objects with the same identity): □quality of being effectively what one claims to be□and □that one thing is the same as another□(Littré dictionary, 1976). The very concept of *set* (in the sense of set theory) is conceived as a unit derived from an original *collection* that may have multiple identities. □I call such multiplicities infinite or inconsistent multiplicities (unconceivable as a unit, a completed object) (□). Conversely, if all the elements of a multiplicity can be thought of as existing simultaneously, in the

[43] Reflecting, for example, the division of objects into components that can change in relation to one another (bills of materials, interfaces, etc)

[44] Mathematically speaking, an □internal composition□law [(re)composition of subset pairs in a nonempty set]

[45] A choice can operate with a function defined across a set of sets and link one of its elements with each and every one of them.

[46] This designation includes the system architecture (subsystems, interfaces, etc.) [Le Moigne, 1977]

[47] Circa -350 bc.

sense that it can be conceived as one single object, I call it a consistent multiplicity or set"[48] (Levy, 1987). The concept of "multiform" does not fall into the category of sets but that of collections, which explains the lack of full interchangeability between "multiform" configurations. "I call set a collection to which we ascribe a concept in such a way that the arrangement of parts[49] is indifferent (in which nothing essential is changed for us when the arrangement alone is altered); and I call plurality A a set all of whose subsets are considered as units of a particular type A, that is, as objects encapsulated by a concept A" (Bolzano, 1993). By generating "multiform" objects, designers contribute to differentiating between *collections* and *sets*. From the perspective of sets, these paradoxical concepts - since they are configurations regarded as equivalent but dependent on the respective contexts of use – crop up when the pace of changes accelerates and the renaming process is unworkable (see 1.2). If interchangeability is relative, contextual and thus conditional, fixed names conceal the relations allowed between subsets and sets, and the choice (in particular the physical choice of objects) becomes unmanageable.

The equivalence of forms that are not discernible by their names ("multiforms") qualifies the classic equality known as "the indiscernibles" according to Leibnitz: "(…) they are the same things of which one can be substituted with the other without compromising the truth" (Leibnitz, 1714, 1998). This equality is total. Two distinct sets cannot have the same elements and two equal sets have the same properties. The naming strategy is thus purely arbitrary: "Statements in a=b form, often have invaluable content for the progress of knowledge and they do not always have an a priori grounding. The discovery that every morning the same sun comes up and not a new sun was certainly one of the most critical breakthroughs of astronomy (…). I use this term [denotation] in the sense of identity and I mean "a=b" in the sense of "a" is the same as b" or "a and b match" (…) The statement a=b may no longer refer to the thing per se but the way we designate it." (Frege, 1879; 1994).

But this relation of equality is not always constructible for designers, including at the mathematical level (Bridges and Reeves, 1999). That is, "(…) to say that a=b if and only if a and b behave the same

[48] Quoted remarks of Georg Cantor

[49] in the sense of subset

in □any context□ (□) is equivalent to the general case [Leibnitz equality]□ (Girard, 2009). Some questions are undecidable, like, for example, the issue of program equivalence (do two given computer programs calculate the same thing?) or the issue of the partial utility of a program (Given that a computer program consists of a set of codes, does it contain a useless subset of codes, that is, never used no matter what use of the program is made?). There is no algorithm (Turing, 1936, 1948; Rice, 1953) that can decide yes or no through a finite number of steps (it is impossible to come up with a method that systematically processes all cases).

The □multiform□ object results from the conception of constructible equivalence classes yet grounded in a relation of equality that is relative more than absolute, and thus more restrictive[50] in the possible contexts (favorable configurations among the combinatory nexus of possible arrangements). An object is distinguishable from another - with which it may or may not be interchangeable □ relative to the known contexts that allow such substitution. This boils down to examining equivalence classes of objects in relation to equivalence classes of contexts. The □relative□ nature of discernibility implies a naming strategy that is broader than the classic arbitrary name in order to represent the possible equivalence classes. □[a=b] cannot be differentiated unless the difference of the signs (a;b) corresponds to a difference in how the designated object is given. (□) Therefore, the statement contains actual knowledge.□ (Frege, 1884).

A class of interchangeable compositions (c_i) and a class of interchangeable use configurations (c_i) make it possible to construct a naming strategy in c_i* | C_i* form, with c_i* and C_i* representing the equivalence classes corresponding to c_i and C_i respectively. c_i* | C_i* literally reads as □any composition of the equivalence class represented by C_i*, knowing[51] that its use is possible in any configuration of the equivalence class represented by C_i*. This c_i* | C_i* form is necessary and contains enough information to

[50] □Polymorphism□ according to Girard P.Y. (2009). Also see Nosofsky R.M. (1984).

[51] In the same sense as the conditionality used by Bayes in statistics [Bernardo J.M. & Smith A.F.M., (1994), Bayesian Theory. Wiley]. Standard naming implies the full interchangeability of use configurations, hence the classic naming strategy: c_i*| with no conditions, and c_i*= c (no name distinction between the collection of fully interchangeable objects and the copy itself).

identify each element and reconstitute the desired subsets and sets (fig. 4). The designers of the firms investigated[52] operated this way, tracking down the invariant elements and those modified between two states of the configuration (initial and final), then developing correspondence models[53] (see 1.3) between the interchangeable compositions (c_i) and the interchangeable use configurations (C_i). This formalism translated in a fully automatable way the relational algebra that designers adopted to absorb the waves of change impacting objects. But since norm-based identification and change rules are usually built around membership relations[54] (families, classes or product categories) rather than equivalence classes, a double correspondence proved necessary.

Fig. 4 Naming strategies

- Standard pattern: initial state (structured sets of named objects) → unchanged names (full interchangeability) or new norm-based names (interruption of interchangeability) → final state
- New generic pattern: initial state (structured sets of named objects) → design of relative equivalence objects → generic names $c_i^* \mid C_i^*$ [or $d^*.R^*$]] → final state

The generic form $c_i^* \mid C_i^*$ echoes the one conceived by E.F. Codd, R.*d* (see 1.4) on the condition that R is interpreted as a representative of the class of equivalent relations (composition, etc.)[55] despite the domain *d* permutations (in which case *d* reads as a representative of the equivalence class of permutable domains). This generic form is compatible with PLM or software engineering tools and DBMSs. Also, it translates the identity of objects into a permanent regeneration of equivalence classes represented by abstraction as Frege (1893) and Fine (2002), for example, understand it (see 1.4).

2.2 Interchangeability, an ability to sustain invariances and design new ones

If we go by a dictionary definition (French Larousse), the term interchangeability first emerged in 1931 to refer to standardized, serially produced parts. The adjective interchangeable predates it, 1870

[52] In particular, within company X's Technical Information System dedicated to spatial activity and armament.

[53] Here referred to as « model program (in-house dictionary of the activity space and defense, Giacomoni G., 1993)

[54] American Standard Code for Information Interchange (ASCII); (NATO) International Standard Organization (ISO); Unified Modeling Language (UML) 2.0 ; System Modeling Language, etc. (see also David F., 2011)

[55] A composition can translate as a relation between compound and component

in France and 1450 in England. It is used for similar same-destination objects that can be changed for or replace one another (interchangeable tires, mechanisms, etc.). Interchangeability has been viewed as *full* or *restricted,* restriction suggesting that, in the limits of known validity, interchangeability can be perfectly and definitively established. Restricted interchangeability □without necessarily having the exact similar content □appears to have emerged in various fields, in biology (Micklem and al; 1975), statistics, in the form of proportion tests[56] of interchangeable values (and models) (Jerdack and Pranab, 1990), mathematics applied to manufacturing (Steiner and Stephenson, 1990) and industrial data processing (Dirickersbach, 2008). *Relative* and *conditional* rather than full interchangeability (Wiggins, 2001; Geach, 1980; Giacomoni, 2002) is a deep recasting of the identity and transformation of technical objects. To contend that any *interchangeability* is relative to existing standards and knowledge (Simon, 1976) and cannot be timeless, boils down to separating the concept of *interchangeability* from the only property that it has ever been credited with; □*to be absolute*□ This holds true for the relation of equality, always taken for granted, no matter if it is full or restricted, even though it is not always constructible in any □context□

There are two types of interchangeability to single out: The element is invariant in a different set (first type). This applies equally to an element common to many unrelated objects and an element common to many generations of one same initial object. The element changes and the set is invariant (second type). These two types translate into distinct contexts of use:

- Use of identical elements in sets (objects) that differ in design and production for first-type interchangeability.
- Use of elements that differ in design and production for second-type interchangeability.

Also, they each have their own application and history. In the first half of the 15th century, printing combined the printing press[57], modeled after screw presses, and movable metal types.[58] Printing

[56] non parametric

[57] which already existed with the □print press□used in wood engraving (woodcut)

[58] which existed as far back as the 16th century (probably as early as the 2nd century), in lead, then in copper) Bi Sheng (1041-1048).

(Chapell, 1970; McMurtie, 1942) essentially built upon other inventions[59] of the time, in particular in the field of the metallurgy (fabrication of punches, casting). The instrumental breakthrough of Gutenberg[60] (Updike, 1920; Scholderer, 1970; Lehman-Haupt, 1966) is the cast and reusable type that alone does away with the requirement of “pressing”. In order to offset casting irregularities (Bertrand, 1787), it was necessary to apply heavy pressure, which marked the back of paper and foreclosed printing. Types remedied the problem of second-type interchangeability (fig.5 appendix 2): standardization of printed editions[61]). In the late 15th and early 16th century, horologists adopted a new drive system, the elastic power of the spring replacing the weight hanging on a pulley. The timekeeping instrument thus became portable. This innovation raises the question of the reuse of the other elements that are still invariant (first-type interchangeability – fig.6 appendix 2). In order to offset the power variations of the spring, horologists used a specific grooved cone named *fusee* whose screw thread was difficult to build manually. Manual cutting did not make for precise spacing or sharp cutouts. J. Ramsden[62] (1735-1800) developed a lathe that made it possible to build a master-screw (leadscrew) required for his high-precision *dividing engine*[63] (1787), later reused for numerous scientific instruments, including sextants, barometers, microscopes, telescopes, etc. The clock is the first modern measuring device and clock makers are the first makers of scientific instruments, pioneering machine-tool technology[64], particularly in the area of the gear, the spring and the screw. The contexts of use of these elements on different products relate to first-type interchangeability. Horology thus became the “mother of machines”, opening up various realms of knowledge and craftsmanship. Printing and horology both presided over a transition from manual productions (manuscript, manual size of the gearwheels) to partially automated productions (cast movable types, dividing engine). The technological breakthroughs (screw, spring, gear) in horology spread to

[59] Including paper (which emerged in China in the early 2nd century) and ink that permits printing on both sides of paper.

[60] A trained goldsmith

[61] The first one is the Mainz Psaulter (1457). Besides, the editions are hybrid, printed and handwritten.

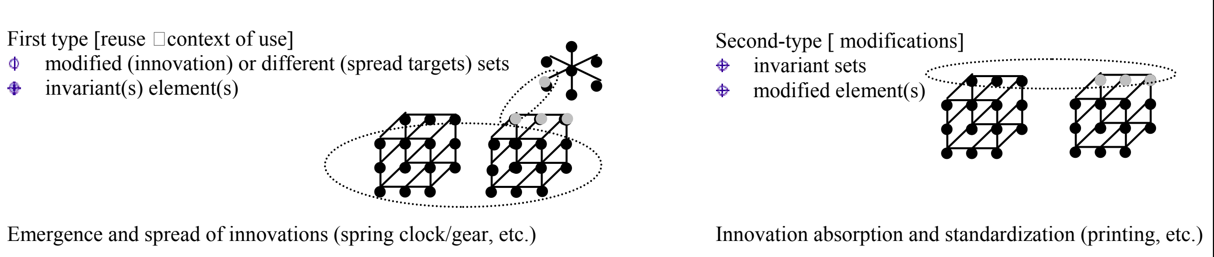
[62] High precision instrument maker who drew upon the techniques of the first French horologists (Lenoir, fusee engine, 1741)

[63] A machine for dividing mathematical instruments

[64] The first gear-cutting machine was designed by J. Torriano (1540) to build a large astronomical clock (20 years in development, 1800 gears and 3,5 years in the making)

scientific instruments. Printing embraced and perfected the technological breakthroughs (die, mold) in metallurgy. Besides, interchangeability marks the ability to sustain invariances and design new ones in terms of emergence and spread (first type), absorption[65] (Saga and Zmud, 1996) and standardization[66] (2nd type). Many industrial transformations have been shaped by the absorption of technological innovations and the standardization of techniques and methods (Cohen, 1994). Basically, serial production requires some technological stability that is questioned by innovation. The following figure (fig.7) details the development process of objects based on an appreciation of the impacted elements (described as second-type interchangeability) or the invariant elements (described as first-type interchangeability). Whether it affects movable and interchangeable types for printing out the pages of a book or interchangeable gears between a clock and a microscope, the interruption of interchangeability signals the end of the duplication[67] of identical and transposable objects (Huang, 1996). This leads to a diversification and segmentation of series as well as potential problems in the management formalism (Hatchuel and Sardas, 1992). It should be noted that elements experience different contexts when used in changing objects, objects experience different contexts when an innovation spreads out, and contexts and contents are relative and vary sometimes separately sometimes simultaneously.

Fig.7 - Interchangeabilités de première et seconde


First type [reuse □context of use]
modified (innovation) or different (spread targets) sets
invariant(s) element(s)
Emergence and spread of innovations (spring clock/gear, etc.)
Second-type [modifications]
invariant sets
modified element(s)
Innovation absorption and standardization (printing, etc.)

[65] Absorption of an external innovation (exogenous) by an environment. The ability of the environment to absorb a technology and develop innovative usages largely hinges on the knowledge of the trade (and thus of use cases) more than the technology per se, as R. Zmud clearly demonstrates. The proposed diagram permits a dual interpretation in terms of knowledge.

[66] NATO & ISO Standards: « The development and implementation of concepts, doctrines, procedures and designs in order to achieve and maintain the compatibility, interchangeability or commonality which are necessary to attain the required level of interoperability, or to optimize the use of resources, in the fields of operations, material and administration. »

[67] □Rule□Design for Assembly n°3: standardizing components [reducing the type of component]

How to control the identity (equality) of objects to prevent interchangeability interruptions that may transform the face of the activity and incapacitate any configuration change management strategy? How to purposely design objects by anticipating their replacement and better absorb the effects of innovation?

2.3 Configuration change management strategies optimized according to redesign and duplication dynamics

We have seen that interchangeability is an invariance of contexts (configurations) or contents (2.2). It is a relative and conditional identity (2.1). We have so far analyzed objects whose product breakdown structure was supposedly known. But we also need to analyze from the perspective of variable breakdown caused by redesign and duplication rates. An overall definition of the object, with no autonomous definition of each of its elements, may be suitable on the condition that this definition is static. On the other hand, as we have seen, if the definition changes due to at least one element changing, the duplication of the object will be problematic as the invariant and the modified elements are totally tied together into one monolithic definition. It will be timely to break down the overall definition so that each constituent element has its own definition and can thus evolve autonomously. The variable elements should be separated and distinguished from the invariant elements. Separating an element involves providing it with an autonomous existence (own identity, autonomous definition and duplication).

♦ *[First] Preservation principle of invariances through separability of variable elements (or departitioning principle)*

The partitioning principle manifests itself, for example, in the design of software/hardware systems, through the hardware[68] (invariant)/software[69] (variable) division with multilayer breakdown, or else

[68] Processors, specific components, memory, complex communication network, etc.

in the division between the design of communication layers and the hardware and software sections (Rousseau, 2005). This principle may make it possible to duplicate the invariant elements by achieving economies of scale (and thus to produce serially) as well as limit the segmentation of series in a diversified economy. The relationships within the system and between the system and its environment can be susceptible to changes affecting all or some of the system or its environment. The equivalence relations (invariance or interchangeability) make it easy to partition a system by bringing together the interconnected entities in order to build self-contained equivalence classes (two classes having one common entity must merge). The departitioning principle aims to prevent any interruption in the known equivalence relations between one entity and the others when its state is likely to change. The modified part is then separated from the invariant part. But the appreciation of invariance and its boundaries relates to the nature of the changes that must occur as well as to the nature of the equivalence relation that already exists with the other entities. Knowledge of the system and its environment determine the application of this principle, which also entails subdividing expansively and as far and often as redesign and duplication rates require. The question of invariance (or interchangeability) can be phrased in two ways:

- How to recognize (and thus identify) the elements (or subsets) that remain interchangeable despite internal changes (second-type), that is, leaving all other elements of their environment invariant when combined[70] with them (functional relation, etc.)?
- How to recognize (and thus identify) the elements (or subsets) that remain interchangeable despite changes in their environment (first type), that is, those that all the other elements of their environment leave invariant when combined[71] with them (functional relation, etc.)?

This leads to the principle of constructability of new invariances with an adequate language and algebra:

[69] Drivers, interrupts, resource management, software/hardware interfaces, application software, etc.

[70] Principle of neutral element of an internal composition law

[71] Principle of stabilizer set

♦ *[Second] constructability principle of invariances through the conception of a language extendable to relative identities (or repartitioning principle)*

This principle states that designing new equivalence classes between elements or subsets, either between the successive generations of one same object or between unrelated objects but having incorporated identical elements (or subsets) throughout their lifecycle often in the wake of an innovation requires redefining the prerequisites of interchangeability in order to recognize and identify the elements (or subsets). The name and relation management strategies are designed and specifically geared toward handling configuration changes. The identity of objects was thus defined using a generic naming language (2.1). In that way, it is no longer defined regardless of the objects life, as is usually the case, but based on the record of compositions and use configurations, which respectively stem from internal and environmental changes. The identity history of an object makes it possible to better design its potential future. As JP Changeux and A. Connes (1989) observe, Since a language is designed to reproduce ( ), it also has a predictive character.

The constructability of invariances arises in relation to context changes that bring on variabilities as well as to the broadening or, more accurately, the context extension. The term extensionassumes the same meaning as in mathematics when a space of objects (or data) is embedded into a larger space. The new prerequisites of interchangeability do not eliminate the preexisting knowledge that helped define the initial prerequisites of interchangeability. For example, the space N of integers is embedded into the larger Q space of rational numbers. The switch (1.4) arises from the collections of the Cartesian product NxN and all the numbers in Q are redefined as ordered pairs (a,b) in a fractional relationship ($\frac{a}{b}$) with one nonnegative denominator b. The integers are reconstructed as ordered pairs (a,1) around this new pattern and have two coexisting names[72] to refer to the same entity, a [relative to N] and (a,1) relative to Q. The identity (interchangeability) of number is redefined in Q: two rational numbers, namely two ordered pairs of integers (a,b) and (c,d) with b and d nonnegative, are equal if

[72] the former is present in the formulation of the latter.

and only if a.d=b.c. Indeed, (2.1) is identical to (4,2) or (8,4), and so forth. In Q, the name (2,1) represents the equivalence class of the ordered pairs generated in this way. Frege (1893) and Fine (2002) use the term abstraction to refer to extension when the objective is to design new equivalence classes that are still unnamed (1.4). Extension involves all objects[73] that are not numbers (Hatchuel and Weil, 2007) and abstraction is performed, for example, in software development (Brunet, 1991; Lee and Al;, 1992; Brönnimann and Al; 2009).

The conception of invariances, like the constant adjustment of configuration change management strategies to the combined dynamics of redesign[74] (innovation) and duplication[75] (serial production), obviously hinges on human collectives (and thus organizations) operating throughout the systems lifecycles (Carbonel, 2001; Patout, 2001; Hussenot, 2007). A great deal of research (Chandler, 1962, 1977; Woodward, 1965; Perrow, 1967; Lawrence and Lorsch, 1967; Giard, 2003; Bourdon and Lehmann-Ortega, 2007) has updated the relationship of systems and organizations with the changes of the environment. The two principles stated above can help better understand the logic of adjustment of configuration change management strategies to the distinctive features of the activity.

The complexity of configuration change management affects sets and essentially depends on three interacting factors: the pace of duplication (series), the pace of redesign (updating of definitions) and lifecycles[76]. As a result, the memorization[77] of necessary and adequate configurations (intended to be duplicated or sustained) in a timeframe that depends on lifecycles) requires to be optimized by

[73] For example, the brand extension as a product differentiation instrument. The brand, according to American Marketing Association is a name, term sign, symbol, design, or a combination of these purposed to identify the goods or services of one seller or group of sellers and so differentiate them from those of competitors. (Ladipo P.K.A., Olufayo T.O. and Omoera C.I., (2012), The Multi-Dimensional Application/Use of Branding in the Universe of Marketers, *International Journal of Academic Research in Business and Social Sciences*, Vol.2, N°4.

[74] Changes stem essentially from enhancements (quality, productivity, etc.) and innovations.

[75] This can be viewed as a material or immaterial production of objects, object definitions, or knowledge.

[76] With long cycles, several generations of objects are destined to coexist if definitions change. Single-use or maintenance-free objects have a limited life cycle at the production cycle.

[77] In all its dimensions: recording, storage, search, access, security, etc.

suitable name and relation management strategies. For example, it is easy to understand that short-run, frequently replaced objects go through increasingly varied[78] definitions. Thus a population of such objects interconnected by an assembly or graph structure generates a combination nexus of possible configurations. Beyond a certain pace of duplication, the redesign frequency eventually requires resorting to the primary principle of invariance preservation through separability of variable elements (to avoid having to redefine objects entirely). A concurrent requirement is the second principle of invariance constructability through the design of a name language extendable to conditional and relative identities. Basically, the ever-faster pace of duplication and replacement ultimately generates subdividable and reconfigurable objects that require an appropriate relations management strategy (including variants, versions, applicability, generation tools, etc.) relayed by a management strategy of absolute, relative or conditional names (interchangeability, modularity, continuous integration, etc.). Accordingly, the relations and names management strategies must remain in line with the industrial dynamics integral to innovation and serial production to avert increasing risks, growing costs and extended lead times.

Conclusion

We tackled the issue of how to manage technical data changes in the contexts of serial production and fast-changing objects as well as their related knowledge systems (which provide the foundations for formal representations). We demonstrated that the issue of managing multiple and repeated impacts on configurations entails theoretical and practical difficulties that cannot be solved in an automated way by tying together PLM functionalities. The high variability of contents (compositions) and contexts (use configurations) spawn polymorphous objects with paradoxical identities. Strategies and formalisms must be able to provide individual or simultaneous variability of identities and relations, based on the industrial dynamics encountered. These questions touch on the conceptual foundations of identity and discernibility, oneness and multiplicity, changes and invariances. They appear in the areas

[78] Encompassing the possible newly generated elements by virtue of the first principle of separability of variable elements.

of software engineering as well as database management systems. We proposed an automatable[79] method for processing these polymorphous objects through a new conception of object identity □ relative and conditional □ which translates composition equivalences based on use configuration equivalences. We also proposed general principles for matching the possible strategies of configuration change management with generic industrial settings variably combining redesign (innovation) and duplication (serial production). Of course, there are several avenues to build on this work, including:

♦ Implementing instrumentally the proposed method. We have made strides in this area, particularly regarding the spatial industry and health care (Giacomoni, 2002; Giacomoni and Sardas, 2011). This experiment validated the concepts and their operational IT implementation. The next step may involve working with or from current PLM systems (the main solutions in the market have been examined) to figure out how this method can fit into the functionality offering and probably continue to work on the choice and interconnection of the various functionalities.

♦ Reflecting on and testing the process from the perspective of organizational implications. Any modeling embedded in an information system is not valuable unless it is based on the skills and needs of users, and there are many of them. The stakeholders fall into two categories:

- Technical data administrators, experts who configure the system from the provided functionalities. One must ensure that these functionalities meet their needs as well as their rationales even if these may change in the context of new procedure trainings.
- The various operators, designers, manufacturers, etc. who need to familiarize themselves with the overarching logic as well as the underlying modeling principles in order to adequately interpret their role as purveyors of information and validation of system-generated outcomes.

All in all, there is progress to be made on the materialization implications at an instrumental and organizational level, keeping in mind that well thought-out automation should adequately include experts both for the design of applications and the control of their operation.

[79] Applied as such in the firm studied (spatial, defense, aeronautics)

BIBLIOGRAPHY

▪ Abramovici M., (2007), "Future Trends in Product Lifecycle Management", in *The Future of Product Development*, Springer Berlin Heidelberg,

▪ Amann K., (2002), "Product lifecycle management: empowering the future of business", CIM Data, Inc.

▪ Batenburg R., Helms R. & Versendaal J., (2005), "The maturity of product lifecycle management in Dutch organizations: A strategic alignment perspective", PLM05: *International conference on product life cycle management,* Lyon, France,

▪ Bellagio D. E. & Milligan T. J., (2005), Software Configuration Management Strategies and IBM

▪ Benbya H. & Meissonier R., (2007), "La contribution des Systèmes de Gestion des Connaissances au développement de nouveaux produits: étude de cas d'une entreprise du secteur de l'industrie aéronautique". *Systèmes d'Information et Management*, Vol.12, N°1,

▪ Bernard A., (1996), *Contribution à la modélisation des produits et à lintégration des technologies productrices dans un environnement multi-acteurs*, HDR, Université Nancy I.

▪ Bertrand J.E., (1787), *Descriptions des arts et métiers, faites ou approuvées par Messieurs de lAcadémie Royale des sciences de Paris. Avec figures en taille-douce*. Nouvelle édition, tome XV, Neuchâtel,

▪ Boothroyd G. & Dewhurst P., (1983),"Design for Assembly", University of Massachusetts,

▪ Bouikni N., Rivest L. & Desrochers A., (2008), "A Multiple Views Management System for Concurrent Engineering and PLM, *Concurrent Engineering: Research and Applications*, Vol.16, N°1,

▪ Bourdon I. & Lehmann-Ortega L., (2007), "Systèmes d'information et innovation stratégique: une étude de cas", *Systèmes d'Information et Management*, Vol.12, N°1,

▪ Bridges D. & Reeves S., (1999), "Constructive mathematics, in theory and programming practice", *Philosophia Mathematica*, Vol.7, N°1,

▪ Brönnimann H., Fabri A., Giezeman G.J., Hert S., Hoffmann M., Kettner L., Pion S., & Schirra S., (2009), *CGAL User and Reference Manual: All Parts,* Release 3.5, October,

▪ Bronsvoort W.F. & Noort A., (2004), "Multiple-view feature modeling for integral product development", *Computer-Aided Design*, Vol. 36,

▪ Brown J., (2006), "Managing Product Relationships: Enabling Iteration and Innovation in Design" *Business Value Research Series*, June, AberdeenGroup.

▪ Brunet J., (1991), "Modeling the World with Semantic Objects", in *Object Oriented Approach in Information Systems*, Van Assche F., Moulin B. & Rolland C. (Ed.), Elsevier Science Publishers B.V. (North Holland),

▪ Cantor G., (1883), *Mathematische Anllalen*, XXI,MiIner J.C. (tr.),

▪ Cantor G., (1895), "*Beiträge zur Begründung der transfiniten Mengenlehre*", in *Mathematische Annalen*, vol. 46,

▪ Carbonel M., (2001), "Dérives organisationnelles dans les projets ERP: les cas de Guerbet et Gaumont", *Systèmes d'Information et Management*, Vol.6, N°1,

▪ Chandler, A.D. Jr., (1977), *The Visible Hand: The Managerial Revolution in American Business,* Cambridge, Harvard University Press,

▪ Changeux J.-P. & Connes A., (1989), *Matière à pensée*, Odile Jacob (ed),

▪ Chapell W., (1970), *A Short History of the Printed World*, Hartley & Marks Publishers Inc., Canada,

▪ CIM Data, (2002), □étude de l□offre SAP pour le cPDm:my SAP PLM, Pilotage des nouveaux produits et processus grâce à la gestion du cycle de vie des produits.□, SAP.AG, Allemagne,

▪ Codd E.F., (1990), □The relational model for database management□, version 2, by Addison-Wesley Publishing Company, Inc.

▪ Codd E.F., (1970), □A Relational Model of Data for Large Shared Data Banks□, CACM 13, N°6, June, Information Retrieval P. Baxendale, Ed.,

▪ Cohen Y., (1994), "Inventivité organisationnelle et compétitivité: l'interchangeabilité des pièces face à la crise de la machine-outil en France autour de 1900", *Entreprises & Histoire*, N°5, juin,

▪ David P., (2011), "MultilingualWeb-LT: Meta-data interoperability between Web CMS, Localization tools and Language Technologies at the W3C", World Wide Web Consortium (W3C), University of Limerick, Dec.,

▪ Demoly F., (2010), *Conception intégrée et gestion d'informations techniques: application à l'ingénierie du produit et de sa séquence d'assemblage*, Thèse, Université de Technologie de Belfort-

Montbeliard, Gomes S. (Dir.)

▪ Dirickersbach J.T., (2008), *Supply Chain Management with SAP APO*, Springer,

▪ Djezzar L., (2003), *Gestion de configuration*, Dunod, Paris,

▪ Eicher C., Hatchuel A., Kieffer J.-P., Molet H., Sardas J.-C., (1984), *Mise en oeuvre et réalités de la Gestion de Production Assistée par Ordinateur*, Editions du CESTA-AFNOR, Paris

▪ Fodor X., (2008), "Organiser les échanges de données techniques: ARIANESPACE ne travaille pas dans la précipitation", Dossier: Collaboration: Optimisez votre travail, *iTechnologie*, N°2.

▪ Feeney M., (1999), *Digital Culture: Maximising the Nation□s Investment: a Synthesis of JISCO/NPO Studies on the Preservation of Electronic Materials*, Londres, National Preservation Office,

▪ Fine K., (2002),*The Limits of Abstraction*, Oxford University Press,

▪ Frege, G. (1994), *Ecrits logiques et philosophiques*, trad. Imbert C., Ed. du Seuil,

▪ Frege G., (1879), *Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*, Halle a.S.: Nebert L., Bauer S. Mengelberg (tr.), (1879-1931), *Concept Notation: A formula language of pure thought, modelled upon that of arithmetic*, in van Heijenoort J., *From Frege to Gödel: A Sourcebook in Mathematical Logic,* Cambridge, MA: Harvard University Press,

▪ Frege, G., (1884), *Die Grundlagen der Arithmetik: eine logisch-mathematische Untersuchung über den Begriff der Zahl*, Breslau: Koebner w., Austin J.L. (tr.), (1974), *The Foundations of Arithmetic: A Logic-Mathematical Enquiry into the Concept of Number*, Oxford: Blackwell, 2nd ed.,

▪ Frege G., (1893-1903), *Grundgesetze der Arithmetik*, Band I/II, Jena: Verlag Pohle H., Furth M. (tr.), (1964), *The Basic Laws of Arithmetic*, Berkeley: U. California Press, vol.I/II,

▪ Geach P., (1980), *Logic Matters*, University of California Press,

▪ Ghoul S., (1983), *Base de données et gestion de configurations dans un atelier de génie logiciel,* I.N.P., Grenoble, Thèse de doctorat d'ingénieur,

▪ Giacomoni G., (2002), "Gérer la reproduction d□objets complexes dans un contexte d□innovation permanente: le cas de l□industrie de l□espace", Thèse de Doctorat, ENSMP, Paris,

▪ Giacomoni G. & Sardas J.-C., (2011), "Pilotage des productions d□objets complexes dans l□industrie de l□Espace : innovation intensive et sériation", *Revue Française de Gestion Industrielle*, Vol.30, N°1,

▪ Giard V., (2003), *Gestion de la production et des flux*, 3e édition, Economica,

▪ Girard J.Y., (2009), "Identité, égalité, isomorphie; ou ego, individu, espèce", d'après un exposé à la réunion LIGC opus 10, Firenze, villa Finaly, 18 Septembre 2009.

▪ Godement R., (2001), *Analyse mathématique*, Vol.1. Convergence, fonctions élémentaires 2ème édition corrigée, Springer-Verlag, Berlin, Heidelberg New York,

▪ Gomes S. & Sagot J-C., (2002), "A concurrent engineering experience based on a cooperative and object oriented design methodology" in Best papers book from 3rd *International Conference on Integrated Design and Manufacturing in Mechanical Engineering*, Kluwer Publishers,

▪ Grieves M., (2006). *Product Lifecycle Management: Driving the Next Generation of Lean Management.*, New York, NY: McGraw Hill,

▪ Hatchuel A., Sardas J.C. & Weil B., (1988), "La mise en uvre et le pilotage d'une GPAO: à chaque étape ses difficultés", *Revue de l'AFGI*,

▪ Hatchuel A. & Sardas J.C., (1992), "Les grandes transitions contemporaines des systèmes de production: une approche typologique", in Dubois M., de Terssac G. (dir.), *Les nouvelles rationalisations de la production*, CEPADUES.

▪ Hatchuel A. & Weil B., (2007), "Design as Forcing: deepening the foundations of C-K theory", Paper submitted at ICED 07, Paris,

▪ Howse D., (1980), *Greenwich Time and the Discovery of the Longitude*, Oxford University Press; 1st Ed.,

▪ Huang Q., (1996), *Design for X: concurrent engineering imperatives*, Chapman & Hall (Ed.),

▪ Hussenot A., (2007), "Dynamiques d'appropriation organisationnelle des solutions TIC: une approche en termes de démarches itératives d'appropriation", *Systèmes d'Information et Management*, Vol. 12, N°1,

▪ Hwang J., Mun D. & Han S., (2009), "Representation and Propagation of Engineering Change Information in Collaborative Product Development using a Neutral Reference Model", *Concurrent Engineering: Research and Applications*, Vol.17, N°2,

▪ IBM, Dassault Systèmes, (2008), "IBM and Dassault Systèmes: Business Process Accelerators for Systems Engineering Integrated Product Development from needs identification through to final product validation".

▪ Jech T., (1978), *Set theory*, Academic Press,

▪ Jerdack G.R. & Pranab K.S., (1990), Annals of the Institute of Statistical Mathematic, Volume 42, Number 1/mars 1990, Springer Netherlands,

▪ Kosman L.A., (1969), Aristotle s Definition of Motion, *Phronesis*,

▪ Lawrence P. & Lorsch J., (1967), *Adapter les structures de l'entreprise - Intégration ou différenciation*, trad. fr., Paris, Editions d'organisation,

▪ Lee S.P., Brunet J. & Rolland C., (1992), "Abstraction in the O* Object-Oriented Method", IFCPAR, Proceedings Indo-French Workshop on Object-Oriented Systems, nov., Goa India,

▪ Lehmann-Haupt H., (1966), *Gutemberg Master of the Playing Card*, New Haven, Yale University Press,

▪ Le Moigne J.-L., (1977), *La théorie du Système Général, théorie de modélisation*, P.U.F., Paris, 1977, 3ème édition mise à jour.

▪ Leibnitz G.W., (1714), *Monadologie*, Trad. Parmentier M., Paris, Vrin,

▪ Leibnitz G.W., (1998), *Recherches générales sur l analyse des notions et des vérités*, Epiméthée, PUF, Paris,

▪ Littré E., (1976), *Dictionnaire de la langue française*, tome III, éditions Famot, Genève,

▪ Mantripragada R. & Whitney D.E., (1999), "Modeling and Controlling Variation Propagation in Mechanical Assemblies using State Transition Models", *IEEE Transaction on Robotics and Automation*, Vol. 15, No. 1,

▪ Marciniak R. & Rowe F., (2008), Systèmes *d'information, dynamique et organisation*, Economica

▪ McMurtie Douglas C., (1942), *The invention of Printing. A Bibliography*, Rochester, N.Y.: Publishing And Printing Dept. Of The Rochester Athenaeum And Mechanics Institute For The Educational Commission Of The International Association Of Printing House Craftsmen,

▪ Merminod V., Mothe C. & Rowe F., (2009), "Effets de Product Lifecycle Management sur la fiabilité et la productivité: une comparaison entre deux contextes de développement produit." *M@n@gement*, 12(4),

▪ Merminod V., (2007), "TIC, Partage de connaissances et fiabilité du développement produit distribué: une approche par le "glitch" au sein du Groupe SEB", *Systèmes d'Information et*

Management, Vol. 12, N°1,

▪ Micklem H.S., Ford C.E., Evans, E.P., & Ogden D.A., (1975), Compartments and cell flows within the mouse haemopoietic system. I. Restricted interchange between haemopoietic sites, Cell Tissue Kinet., May, Vol.8 N°3,

▪ Mostefai S. & Batouche M., (2005),"Data integration in Product Lifecycle Management: an ontology-based approach", PLM05: International conference on product life cycle management, Lyon, France.

▪ Neagu N. & Faltings B., (2001), Exploiting Interchangeabilities for Case Adaptation, Lecture notes in computer science, Springer, Berlin, Allemagne, Etats-Unis,

▪ Noël F., (2006), "A dynamic multi-view product model to share product behaviors among designers: how process model adds semantic to the behavior paradigm", *International Journal of Product Lifecycle Management*, Vol.1, N°4,

▪ Nosofsky R.M., (1984), "Choice, similarity, and the context theory of classification", *Journal of Experimental Psychology*: Learning, Memory and Cognition; vol.10, n°1,

▪ Patout Y., (2001), "La réorganisation de la téléphonie par les profils", *Systèmes d'Information et Management*, Vol. 6, N°1,

▪ Perrow C., (1967), "A Framework for the Comparative Analysis of Organizations", in *American Sociological Review*, Vol.32, N°2,

▪ Pol G., Jared G., Merlo C. & Legardeur J., (2005), "From PDM systems to integrated project management systems: a case study", *PLM05: International conference on product life cycle management,* Lyon, France.

▪ Ramsden J., (1787), *Description dune machine pour diviser les instruments de mathématiques*, Londres, Bureau des Longitudes, traduite de lAnglois, Paris,

▪ Raymond E.S., (1998), *The Cathedral and the Bazaar*, OReilly,

▪ Rehman F.U., Yan X.-T., (2007), "Supporting early design decision making using design context knowledge", *Journal of Design Research*, Vol.6, N°1-2,

▪ Reix R., Kalika M., Fallery B. & Rowe F., (2011), Systèmes d'information et management des organisations, 6e éd., Vuibert,

▪ Rice H.G., (1953), "Classes of recursively enumerable sets and their decision problems", *Trans. Am. Math. Soc.*, vol. 74, n°2,

▪ Rousseau F., (2005), *Conception des systèmes logiciel/matériel: du partitionnement logiciel/matériel au prototypage sur plateformes reconfigurables*, Thèse d HDR, Université Joseph Fourier, Grenoble I,

▪ Russel B., (1900), *A Critical Exposition of the Philosophy of Leibnitz*, Cambridge, The University Press,

▪ Russel B., (1903), *The principles of mathematics*, New York,

▪ Saga V. & Zmud R. W., (1996), « Introduction de logiciels de gestion dans des petites entreprises liées à une profession libérale », *Systèmes d'Information et Management*, Vol.1, N°1,

▪ Sacquet A. & Nowencien R., (1995), "Productivité de la maintenance en Architecture client-serveur", revue *Génie Logiciel* n°35,

▪ Scholderer V., (1970), *Johannes Gutemberg, The Inventor of Printing,* British Museum, 2nd Ed.,

▪ Simon H., (1976), "From substantive to procedural rationality", in: S. LATSIS, *Method and appraisal in economics*, Cambridge, Cambridge University Press,

▪ Simondon G., (1958), *Du mode d'existence des objets techniques*, Aubier,

▪ Stark J., (2004), "Product Lifecycle Management - 21st century Paradigm for Product Realiza-tion.",*Decision Engineering Series,* Springer Verlag, Berlin,

▪ Steiner G. & Stephenson P., (2000), Subset-Restricted Interchange for Dynamic Min-Max Scheduling Problems, SIAM Journal on Discrete Mathematics, archive Vol.13, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA,

▪ Turing A.M., (1936), "On Computable Numbers, with an Application to the Entscheidungs problem", *Proceedings of the London Mathematical Society*, Series 2, 42,

▪ Turing A.M., (1948), "Intelligent Machinery", National Physical Laboratory Report, in Meltzer B., Michie D., (eds), (1969), *Machine Intelligence,* 5, Edinburgh University Press, Edinburgh,

▪ Updike D.B., (1920), *Printing Types: Their History, Forms and Use*, 2 vol., Harvard and Oxford,

▪ Wiggins D., (2001), *Sameness and Substance Renewed*, Cambridge University Press,

▪ Woodward J., (1965), *Industrial organization. Theory and practic*e, Oxford University Press,

Appendix 1: Excerpts from functional descriptions

[1]□(□) the potential for re-use physical items is determined by using the names and descriptions to identify the similarities and differences between items naming conventions (□) a generic noun is used as the name to describe each physical item (□) each item is described by identifying its attributes in their descending order of significance [Manufacturer part number] (□) form of interchangeability codes (EDI standards) for restricted additional interchangeability relationships (□) user interface (□) serial number (□) indices de modification d□article□
□(□) modification number (□) versions (□) variants (□)□
□(□)Interchangeability of similar items is confirmed by comparing their designs (□) full interchangeability form-fit-function-class (FFF class) (□) restricted interchangeability relationships between groups of fully interchangeable parts (□) sub-item for company-internal technical mapping of a part interchange (□)definition and management of item group hierarchies for locating standard items
□(□) extension of an item division (□) classification (□) configurable vs. non-configurable(□) item (□) bill of material dependencies (□) kernel (which represents a component or function of the product structure) (□) variance diagrams for checking variant completeness and unambiguity)□
□(□) inventory overview for the stocks of interchangeable parts in context (□) cumulative analysis of interchangeable parts (□) ERP links (□) The sophisticated functionality [X] incorporates the change management processes between engineering and production: analysis of changes entailed by conflict search; generation of an update reporting warnings, errors or absence of conflict: search of impacted production orders, creation of related modification requests (□)□
[SAP □PLM]

□(□) standard□
□(□) serialized items (□) use cases and spares (□)□
□ (□) tracking and traceability (□) impact analysis (□) reference configurations (baselines) (□) date or rank-based variants and effectivity (□) version history and configuration comparison (□) error and deviation processing
□ (□) decision-making chain (□) interface with ERP system (□) pre-procurement process and processing of configuration changes connected with purchase and production constraints
[LASCOM □ICS]

---

[10] □(□) standard□
[11] □(□) generic product, platform, variant requirements: systematic approach to organize products (logical set of modules) into interchangeable modules (□) appropriate module interfaces (□)□
[12] □(□) baselines, release level, effectivity (□)□
[13] □(□)translation of requirements into required functions (□) high-level platform architecture design (□) product module interfaces formalized to manage the transfer and reintegration of design data, while preserving its integrity (□) changes control option sets for Assemble-to-Order products (□) options; combinations (□)□
[14] □(□) decision management process; formal change impact and root cause analysis; configuration traceability (□)□
[PTC - Windchill]

---

[15] □(□) serial numbers units (□) alternate parts and acceptable substitutions are communicated and managed to allow for manufacturing flexbility with adequate control for managing quality standards (□)□
[16] □(□) re-use of multiple product variants or product□s evolution; tracking original requirement to the final product (□)□
[17] □(□)modifications for a set of configurations (□) specific bills-of-material views for unique product and/or tracking changes; product configuration alternatives with traceability for design changes, analysis of impacts, detail design, downstream processes and compatible effectivity (...) configured infrastructure linking product design with logical and functional definitions; configured context for product components design and modifications; effectivity ranges for added and modified product components; variants or state of the product (□)□
[18] □(□) a key requirement for engineering change is the approval and notification process (□) viewing and tracking of all changes to all product configurations; program deliverables with reliable and up-to-date information to all stakeholders at all times; investigation of new variation of product with virtual testing and performance-based decision-making; risk management (□)□
[19] This solution is based on the software products CATIA, ENOVIA, DELMIA, SIMULIA
[DASSAULT Systèmes - DS Portfolio]

---

[20] Teamcenter, NX, Tecnomatic, Velocity/Solid Edge
[21] □(□) impacts of a design change managing the relationships between parts, their technical data, and the documentation that supports them (□) data assembly code; identification/status information which includes various kinds of metadata that determines the access rights and configuration controls for the data module; model identification (□)□
[22] "(□) impact analysis; context monitoring; synthesis/comparison for the control of assembly relative to design; comparison of the various bills of materials with the original state; traceability of critical elements(□) implied relationships between product structure and parts data (relative to part numbers, configuration and effectivity))□
[23] □(□) standard numbering system; proven interoperability through a common information platform for integration [content interchangeability and use of context-driven content]; relationships within a subsystem and across the rest of the platform subsystems [to ensure the primary system complies with the most challenging requirements (from concept development through production)]: managing the inaction and impact of all subsystems and ultimately ensuring the collective performance of all subsystems satisfies the overall platform/prime system/vehicle performance requirements (□) commonality and re-use capabilities (□) configuration and change management with digital simulation (□)□
[24] □modular scenario for data management {content/identification + status data (metadata)} necessary for controlling the data module and its configuration; design process and variants module enhancements (links between variant filters and variant sets in design process) (□)□
[25] □(□) fully understand the impact of product changes; collaborative solution for planning and validating the manufacturing assembly processes; evaluate different assembly alternatives, plan for multiple variants and management change across the entire assembly process lifecycle; review and adjustment of the product; multiple decisions about what elements and attributes of the specification need to be used; multiple lifecycle states (□)□
[SIEMENS □Portfolio]

Appendix 2: Printing press (second-type interchangeability) and horology (first-type interchangeability)

Fig.5 Printing press second-type interchangeability

Production method of cast types:

- punchcutting the desired glyph in relief and reverse with the end of a mild steel metal bar
- striking a matrix in a softer metal (red copper) using this letterpunch
- casting (reproduction) an alloy of lead, tin and antimony in a hand mold.
- Placing (words, sentences, pages, etc.) the types upside down on a base.
- Placing a sheet of paper over it.
- Pressing a platen against a seamlessly flat printing block

- *impression* de pages recto verso et reproduction à l identique.

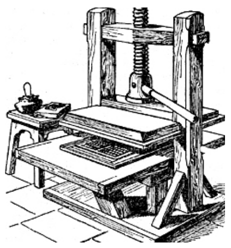
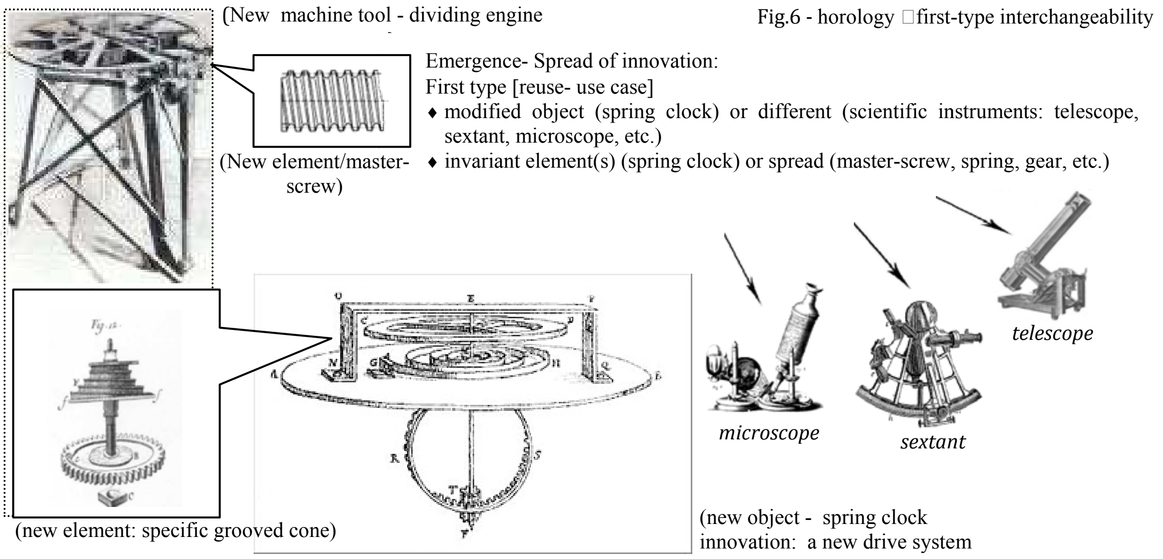


Fig.6 - horology first-type interchangeability


(New machine tool - dividing engine
(New element/master-screw)
(new element: specific grooved cone)
(new object - spring clock
innovation: a new drive system
microscope
sextant
telescope

Emergence- Spread of innovation:
First type [reuse- use case]
♦ modified object (spring clock) or different (scientific instruments: telescope, sextant, microscope, etc.)
♦ invariant element(s) (spring clock) or spread (master-screw, spring, gear, etc.)