Towards Immunization of Complex Engineered Systems: Products, Processes and Organizations

By

Mahmoud Efatmaneshnik

A Thesis Submitted for the degree of Doctor of Philosophy
School of Mechanical and Manufacturing Engineering
The University of New South Wales
January, 2009

ORIGINALITY STATEMENT

'I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged'

Signed ……..

Date ……..

COPYRIGHT STATEMENT

Signed ……

Date ……

AUTHENTICITY STATEMENT

'I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis. No emendation of content has occurred and if there are any minor variations in formatting, they are the result of the conversion to digital format.'

Signed ……..

Date ……

Abstract

Engineering complex systems and New Product Development (NPD) are major challenges for contemporary engineering design and must be studied at three levels of: Products, Processes and Organizations (PPO). The science of complexity indicates that complex systems share a common characteristic: they are robust yet fragile. Complex and large scale systems are robust in the face of many uncertainties and variations; however, they can collapse, when facing certain conditions. This is so since complex systems embody many subtle, intricate and nonlinear interactions. If formal modelling exercises with available computational approaches are not able to assist designers to arrive at accurate predictions, then how can we immunize our large scale and complex systems against sudden catastrophic collapse?

This thesis is an investigation into complex product design. We tackle the issue first by introducing a template and/or design methodology for complex product design. This template is an integrated product design scheme which embodies and combines elements of both design theory and organization theory; in particular distributed (spatial and temporal) problem solving and adaptive team formation are brought together. This design methodology harnesses emergence[1] and innovation through the incorporation of massive amount of numerical simulations which determines the problem structure as well as the solution space characteristics.

Within the context of this design methodology three design methods based on measures of complexity are presented. Complexity measures generally reflect holistic structural characteristics of systems. At the levels of PPO, we correspondingly introduce, the Immunity Index (global modal robustness) as an objective function for solutions, the real complexity of decompositions, and the cognitive complexity of a design system. These three measures are helpful in immunizing the complex PPO from chaos and catastrophic failure.

[1] A property of a system is emergent if and only if the property is present in global scales and cannot be traced to the local properties of parts of the system, an emergent property is thus the global effect of local interactions.

In the end, a conceptual decision support system (DSS) for complex NPD based on the presented design template and the complexity measures is introduced. This support system (IMMUNE) is represented by a Multi Agent Blackboard System, and has the dual characteristic of the distributed problem solving environments and yet reflecting the centralized viewpoint to process monitoring. In other words IMMUNE advocates autonomous problem solving (design) agents that is the necessary attribute of innovative design organizations and/or innovation networks; and at the same time it promotes coherence in the design system that is usually seen in centralized systems.

Acknowledgement

First and foremost, I would like to extend my gratitude to Dr. Carl Reidsema, my supervisor, who has greatly contributed to this work; the presented material on engineering design methodology and Decision Support Systems is much in debt to his knowledge and supervision. His support has been accompanied by smart and yet brave management, which reflects his understanding of postgraduate supervision. The content of this work contains a considerable amount of novelty which is owed to the support, encouragement and mentoring of Dr. Jacek Marczyk.

I am very grateful to the heads of Mechanical School of UNSW, Prof. Kaebernik, and Prof. Leonardi and also the postgraduate research coordinator, Prof. Randall, for showing understanding, support and encouragement. I would like to sincerely thank my previous supervisors: Mr. John Page and Dr. Berman Kayis who gave me the opportunity to undertake this PhD program. Also many thanks go to my ex-co-supervisor Dr. Inge Koch from Mathematical School whose presence gave some depth and insight into parts of this work.

I am and have been always grateful to my parents (Fatemah and Ali), my sisters, and brothers for the support, encouragement, and tolerance during these years. I would like to extend my appreciation to Dr. Asghar Tabatabaie and Hadith for, indeed, being friends in need. Also I would like to thank the Conceptual Design Laboratory students, Luke Djukic, Garth Pearce, Robert Wootton and Ian Watson for being kind and caring companions.

I am grateful to Warrane College staff for bestowing the peace of mind in a supportive environment, especially David Curran and the priest Fr. Anthony Khoudair. Finally, I would like to thank my spiritual guide, Prof. Nader Angha, whose distant call is an everlasting source of hope for those who accept and choose to know.

PUBLICATIONS DURING Ph.D. CANDIDATURE

Efatmaneshnik M, and Reidsema CA. 2007. Immunity as a Design Decision Making Paradigm for Complex Systems: a Robustness Approach. *Cybernetics and Systems*, Vol 38, No 8, pp 759-780.

Efatmaneshnik M, Reidsema CA. 2007. Immunity and Information Sensitivity of Complex Product Design Process in Overlap Decomposition. *InterJournal*, Complex Systems Section, Manuscript No. 2035.

Efatmaneshnik M, and Reidsema CA. 2008. *Decomposition Modes and Integration Schemes in Complex Systems Design.* In proceedings of Systems Engineering Test and Evaluation SETE 2008, September, Canberra.

Efatmaneshnik M, and Reidsema CA. 2008. *Exploiting Non-Dominance in Multi Agent Systems: An Artificial Immune Algorithm for Distributed and Complex Problem Solving Environments.* In proceedings of 12th Asia Pacific Symposium on Intelligent and Evolutionary Systems IES08, 7-8 December, Melbourne.

Efatmaneshnik M, and Reidsema CA. 2009. IMMUNE: A Collaborating Environment for Complex System Design. In *Studies in Computational Intelligence: Collaboration, Fusion and Emergence,* Eds Mumford C, and Jain L, Ch 9, Springer. “in press”

Table of Contents

List of Figures

List of Tables

1 Why Complexity?

The science of complexity has roots both in natural and social sciences (Erdi, 2008). Physicist, Heinz Pagels (1989) regarded complexity as being on the cutting edge of science and stated that:

> I am convinced that the societies that master the new sciences of complexity and can convert that knowledge into new products and forms of social organization will become the cultural, economic, and military superpowers of the next century.

In the same lines on January 23, 2000 Stephen Hawking said in "San Jose Mercury News", "I think the next century will be the century of complexity" (Erdi, 2008). Human history has demonstrated a relentless progress towards greater complexity —in man made products and artefacts (Minai et al, 2006). The last two centuries in particular can be characterized by a radical move towards greater complexity, in economical, political, social and technological systems (Minai et al, 2006) such that complexity, desired or otherwise, is now dominating almost every aspect of modern life (Marczyk, 1999). Today's challenges and future ones necessitate novel approaches to understanding, analysing and synthesizing complex and interconnected large scale systems (Sanders, 2003). For that reason, complex systems science is an imperative for the future of scientific evolution and has captured the attention of scientists from almost every scientific discipline (Ottino, 2004).

This need for novel approaches as part of the science of complexity in engineering is even more obvious since engineered systems are becoming more and more complex and the consequences of increasing complexity are inevitable (Marczyk, 1999). The design, implementation, and manufacturing of new complex products with tight performance and quality requirements, and strict cost constraints contributed to the fragility of the engineering projects (Marczyk, 1999). The design of complex systems with emergent properties (collective

properties absent at the local level of parts) is almost impossible with the traditional engineering tools and methods (Ottino, 2004).

A complex situation is one in which a large number of independent variables interact (Sanders, 2003). According to Erdi (2008) simple systems are characterized by:

1. Single cause and single effect
2. A small change in the cause implies a small change in the effects
3. Predictability

In contrast complex systems have interconnected elements and are characterized by (Erdi, 2008):

1. Circular causality, feedback loops, logical paradoxes, and strange loops[2]
2. Small change in the cause implies dramatic effects
3. Emergence and unpredictability

Complex systems intrinsically possess potential for catastrophic failure since the behaviour of complex systems is not predictable from the knowledge of individual elements, no matter how much we know about them (Merry, 1995; Cook, 2000). Complex systems require special treatment since they possess potential for catastrophic failure (Merry, 1995; Marczyk, 1999). The failure in unsuccessful complex projects (like redesign of the air traffic control systems) is often attributed to simple reasons (Bar-Yam 2003). For example the fatal outcome of the Challenger mission disaster was caused due to failure of a seemingly innocent component; in another case the first unsuccessful launch of Ariane 5 was caused by trivial software problems (Marczyk, 1999). Similar examples include the crashes of new generation commercial aircraft due, in the majority of cases, to minor but unforseen software problems. It is well known that modern flight control avionics systems are extremely complex; what complicates the situation is

[2] Circular causality in essence is a sequence of causes and effects whereby the explanation of a pattern leads back to the first cause and either confirms or changes that first cause (Erdi, 2008).

that the pilot (man in the loop) is a stochastic entry which can occasionally enter into conflict with the flight computer (Marczyk, 1999).

While people have attributed the failure of the advanced automation systems (computer based systems) as the root cause of these kinds of problems, according to Bar-Yam (2003) “the magnitude of failures of the large projects shown in Table 1.1, and the suggestion that each case involved its own unique reasons does not seem to strike at the core of the causes of failure”. Despite the fact that failure in any specific case may be appear to be related to a specific cause, but the *common inability* to implement large scale and high cost systems can be attributed to their “intrinsic complexity” (Bar-Yam 2003). In fact according to Marczyk (1999) “complexity in conjunction with uncertainty brings about a host of new problems and phenomena”. He explained that the complexity principle perceived by Lotfy-Zadeh (1973) reflects the new status quo of science and engineering. This principle states that:

> As the complexity of a system increases, human ability to make precise and relevant (meaningful) statements about its behaviour diminishes until a threshold is reached beyond which the precision and the relevance become mutually exclusive characteristics.

Uncertainty, innocent and harmless in simple systems, becomes a fundamental issue, namely the introducer of fragility in large and complex systems: with the combination of complexity and uncertainty, catastrophe is always around the corner (Merry, 1995; Marczyk, 1999). Casti (1994) describes catastrophe:

> Occasionally, in a system, we encounter a combination of input values such that if we change them only a small amount, the corresponding output will shift discontinuously to an entirely new region. This is called bifurcation and it is considered a catastrophe.

Table 1.1 The magnitude and scale of the failed projects due to their inherent complexity (From Bar-Yam, 2002).

System Function Responsible organization	*Year of Work (Outcome)*	*Approximate Cost*
Vehicle Registration, Drivers license Dept. Of Motor Vehicles	1987-1994 (Scrapped)	$44M
Automated reservations, ticketing, flight, scheduling, fuel deliver, kitchens and general administration-United States Airlines	Late 1960s-Early 1970s (Scrapped)	$50M
State wide Automated Child Support System(SACSS)-California	1991-1997 (Scrapped)	$110M
Hotel reservation and flights-Hilton, Marriott, Budget, American Airlines	1988-1992 (Scrapped)	$125M
Advanced Logistics System-British Stock Exchange	1968-1975 (Scrapped)	$250M
Taurus Share trading systems-British Stock Exchange	1990-1993 (Scrapped)	$100-$600M
IRS Tax Systems Modernization Projects	1989-1997 (Scrapped)	$4B
FAA Advanced Automation System	1982-1994 (Scrapped)	$3-$6B
London Ambulance Service Computer Aided Dispatch Systems	1991-1992 (Scrapped)	$2.5M, 20 lives

The new region to which a system enters may be characterized by higher complexity, implying higher fragility. Bifurcation by itself may not signify failure but it has the potential to introduce tremendous amount of fragility and vulnerability into the system: simply, a bifurcation can change the mode of a system from a robust mode to a potentially non-robust and dangerous mode which would, because of its unpredictability, lead to catastrophic outcomes (Efatmaneshnik and Reidsema, 2007.a). As such, catastrophe is a major reason for the failure of complex systems.

The catastrophic behaviour of complex systems can be, however, discovered by studying how the system elements interact and how the system changes and adapts through time as a result of this interaction (Merry, 1995). As Sanders (2003) has described “in essence complexity science is moving us away

from a linear, mechanistic view of the world, to one based on nonlinear dynamics, evolutionary development and systems thinking." Sanders argued that the insights from complex systems research provide a dramatically novel *theory-driven framework* for understanding and influencing the complex systems, and their emergent properties.

1.1 Complex Engineering Design

Design is about constructing artefacts and the engineering design of artifacts in a mechanical engineering context constitutes the following steps (Finger and Dixon, 1989):

- recognition of need
- specification of requirements
- concept formulation (design synthesis)
- concept selection (design analysis)
- embodiment of design detail

It is important to distinguish between the design/redesign of previously established products and that of completely new ones. *New Product Development* (NPD) is the process of bringing a new product to market e.g. an aircraft, a computer or a Mars probe. The product can be regarded as a system, the common definition of which is set of interacting or interdependent entities, real or abstract, forming an integrated whole[3]. These entities can be systems on their part in which case the term System of Systems is used. A complex product is a system of systems (or subsystems). Systems engineering, in general, is an interdisciplinary field of engineering. It focuses on the development and organization of complex artificial systems and is defined by INCOSE [4] (International Council On Systems Engineering) as:

[3] From Wikipedia, the free online encyclopaedia.
[4] Source: http://www.incose.org/practice/whatissystemseng.aspx.

> A branch of engineering whose responsibility is creating and executing an interdisciplinary process to ensure that customer and stakeholder's needs are satisfied in a high quality, trustworthy, cost efficient and schedule compliant manner throughout a system's entire life cycle, from development to operation to disposal.

This process usually comprises the following seven phases:

1. Stating the problem
2. Investigating alternatives
3. Modelling the system
4. Integrating
5. Launching the system
6. Assessing the performance
7. Re-evaluating

The phases involved in systems engineering are performed in a parallel and iterative manner rather than sequentially: and is often referred to as Concurrent Engineering. Keating et al (2003) explained that “systems engineering integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation". Systems engineering methodologies have been applied to the design of “loosely coupled” systems successfully. The integration problem in Complex NPD and its tasks at all levels of PPO involves a tremendous effort. At the integration level, it has been observed that often complex NPD processes tend to spiral out of control (Mihm and Loch, 2006): the process oscillates between being ahead and behind the schedule. Mihm and Loch (2006) describe a host of these types of oscillating process performance behaviours resulting in failures within various industries; for example, in the development of the Microsoft Office Project (Iansiti, 1990), in the aeronautics industry for the Boeing 767-F project (Klein et al, 2003.a), and from the semiconductor industry in Intel’s Itanium chip design project (Hamilton, 2001). Engineers of future complex systems face an emerging challenge of how to address problems associated with integration of

(multiple) complex systems (Keating et al, 2003). In response to this need a branch of systems engineering is beginning to emerge known and be known as Complex-System Engineering.

Bar-Yam (2003) spoke of the radical departure of complex systems design from traditional systems engineering. Acknowledging that complexity is not a new phenomena in engineering and has been addressed by the traditional systems engineering (for centuries), he argued that for designing highly complex systems, two of the main methods of traditional systems engineering that deal with complexity need to be discarded, namely, abstraction and modularity. Abstraction simplifies the description or specification of the system. Decomposition of a problem/system into modules (or sub-problems/sub-systems) is a well recognized way to separate a large system into parts that can be individually designed and modified. Bar-Yam (2002) added that:

> These methods are useful, but at some degree of interdependence (in the system's elements) they become ineffective: modularity incorrectly assumes that a complex system behaviour can be reduced to the sum of its parts and abstraction assumes that the details to be provided to one part of the system (module) can be designed independently of details in other parts.

Bar-Yam (2002) emphasized the concepts of "radical innovation", "gradual implementation" and "evolutionary engineering" as the remedies for the difficulty of integration within complex systems design. In a more moderate view, Kuras (2007) called for modifications in the traditional engineering process/template as to suit complex systems design. He portrayed a fundamental difference that traditional systems engineering and complex systems engineering ought to have: the need for multiple conceptualizations (evolutionary and dynamic modelling) of the system at different scales as compared to one static conceptualization (unchanged model) of the system in traditional systems engineering. Ring and Madni (2005) acknowledged that a static model of the system is not only insufficient but also leads to serious misunderstandings and

under-conceptualization of the solution, which opens the way for unintended consequences (for example; catastrophic failures). Keating et al (2003) voiced several other modifications to the traditional systems engineering, such as:

- Avoiding optimization based decision making
- Avoiding strict goal and objective setting (anticipating constraint relaxation)
- Maintaining distributed focus during the design process (as opposed to a single focus in traditional systems engineering)

Norman and Kuras (2006) stressed the presence and actions of autonomous agents as important elements in complex systems engineering, since they can significantly contribute to the innovativeness of the design system. Norman and Kuras also asserted that complex systems engineering is not simply an increased attention to detail rather it is an attention to overall coherence. The overall regimen of complex system engineering must create and manage an environment in which multiple autonomous agents each address a fraction of the relationships that might be involved in an overall complex system (Norman and Kuras, 2006).

1.2 Objective: Immunity of Complex PPO

In the context of systems engineering there are three main streams in which complexity can be addressed:

- Complexity of Products
- Complexity of NPD Processes
- Complexity of NPD Organizations

In addition to inherent complexity of product, process and organization, there is another source of complexity in NPD: this complexity arises from the coupling between the product, process and organization structures. This coupling,

depicted in Figure 1.1, makes the planning for development of even not-that-complex products, a complex task.

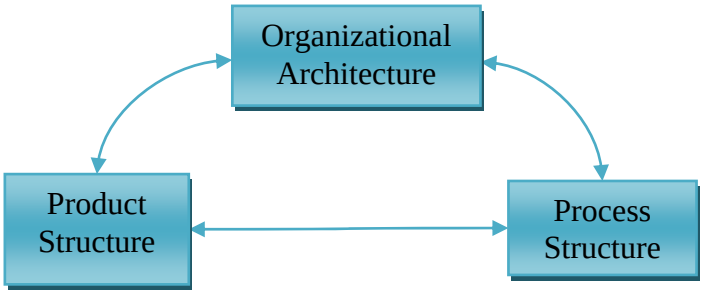


Figure 1.1 Product, process and organization structures are tightly related, after Browning (2001).

Planning and problem solving are discussions in the domain of artificial intelligence. Within artificial intelligence three main sub-domains extensively deal with complexity:

- Swarm Intelligence: characterizes collective behaviour of decentralized, self-organized systems.
- Artificial Life: explores the logic of living organic systems in artificial settings.
- Artificial Immunity: induction of immunity into artificial systems.

These three areas overlap in that they deal with complexity, emergent properties and collectives (Figure 1.2). Artificial immunity in particular seems to have a lot to offer complex systems engineering. After justifying that complex systems are prone to sudden failure and collapse without prior notice, it is enticing to think that complex systems can be immunized. Hart et al (2009) introduces the new concept of Immuno-engineering as:

> The abstraction of immuno-ecological and immuno-informatics principles, and their adaptation and application to engineered artefacts (comprising hardware and software), so as to provide these artefacts with properties

analogous to those provided to organisms by their natural immune systems.

This thesis explores the ways in which complex systems (products, processes and organizations) might be immunized. Some core elements of the immune system such as recognition of self from non-self are employed and are central to the approach of this thesis. It should however be noted that the algorithms presented in this thesis for immunizing PPO, may not seem deeply immuno inspired or exact mimics of computational models of biological immune systems; rather the algorithms may appear as reasoned by the metaphor of immunity. Hart et al (2009) and others (Stepney et al, 2005; Timmis, 2007) see *reasoning by metaphor* as a drawback for Artificial Immune Systems as it does not allow for full exploitation of the field and will limit the ultimate success of the field.

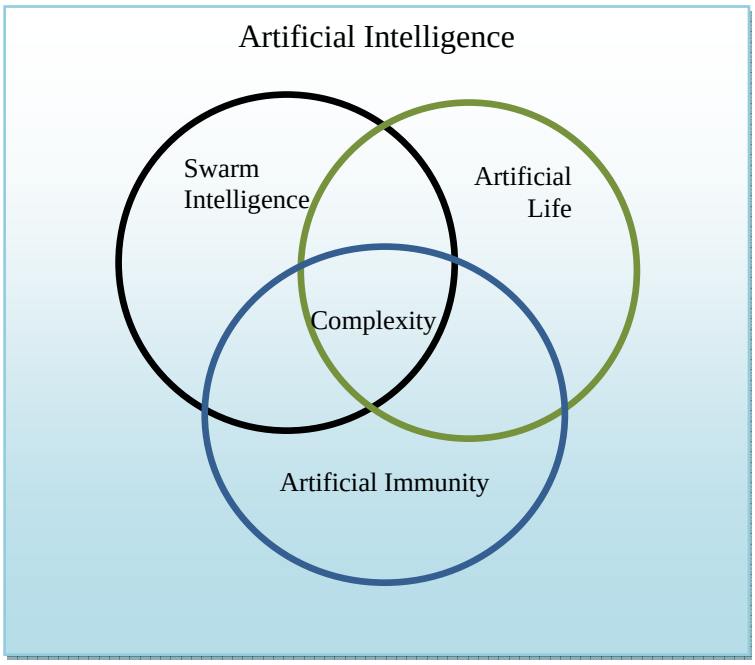


Figure 1.2 Sub-domains of artificial intelligence that address complexity.

While we accept the above limitation to this work, it should be noted that this thesis provides more rigor in its effort to establish a template for complex systems engineering, in that the presented model (methodology) and methods

(algorithms) are computationally justifiable (and not just vague recommendations or imperatives). Furthermore, we believe that the introduction of the immunity metaphor to the engineering community through this work is per se a graceful move in the right direction towards dealing with complex systems. Natural systems are robust because they have immune systems and not vice versa, and this message can have important implications for the future of engineering design research.

Measurement constitutes the basis of any rigorous scientific activity (Marczyk and Deshpande, 2006) and the methods presented here exploit the measures of complexity at all levels of PPO. The contribution of this thesis is summarized in the following:

- Measure theoretic approach to harnessing robustness and simplicity in products.
- Measure theoretic approach to decomposition and integration: the deficiency of decomposition based problem solving is explicitly shown.
- Measure theoretic approach to holistic process monitoring to maintain coherence in multi agent design system.

A background of engineering design methodologies and the necessary definitions for understanding the thesis is presented in Chapter 2. Chapter 3 is about complexity, complexity measures and the literature review of their utilization in engineering design. It ends with a discussion on the methodology of this thesis. Chapter 4 presents our template for complex product design. Chapters 5, 6, and 7 describe three design methods at all levels of PPO and how complexity measures can immunize, or reduce the risk of their, sudden failure. All the material described up to Chapter 7 constitutes the domain knowledge of a DSS which we refer to as IMMUNE and is presented in Chapter 8.

2 Background: Engineering Design

Engineering design is about creating a desired functionality that may be complex and managing the design process of a complex product. Nobel laureate Herbert Simon in the Sciences of the Artificial (1969) proposed that the scientific method can be used to hypothesize and develop a design science. He asserted that design may be regarded as a science more than an art. The aim of science is to *explain* as much as possible the complexity of the natural phenomena and its processes; whereas designers want to *create* complexity in the form of complex artefacts. From this standpoint natural sciences and design science overlap in their extensive use of observations, and predictions. Both designers and scientists seek to find the contributing parameters for the behaviour of the systems and determine the interplay of these parameters. Within this perspective both natural sciences and design science are problem solving procedures.

The problem solving process is a search and decision making process: searching for solutions of a given problem then opting and deciding, from those solutions for the one that has the highest utility (Rusbult, 2000). The process of design can be characterized in terms of its efficiency and effectiveness. The efficiency of a design process is mostly determined by high level characteristics such as the quality of the final product, the design lead time and overall cost (Fixson, 2005). The design process effectiveness, on the contrary, is dependent by how the designers have met the top level goals with their finite resources (Fixson, 2005). In the case of complex systems the design efficiency issue is not a trivial one. To address the concerns about the efficiency and effectiveness of design processes, design methods are developed Design methods are tools for designing that can be integrated into the design process (Cross, 2000). Design methods generally lead to more holistic solutions and thus better final products by gaining important insights about the design PPO. However there is no solid consensus about their validity. Cross (2000) states that:

> It might seem that some of these new methods can become over formalized, or can be merely fancy names for old

> common-sense techniques. They can also appear to be too systematic to be useful in the rather messy and often hurried world of the design office. For these kinds of reasons, many designers are still mistrustful of the whole idea of design methods.

The main argument against design methods is that they might destroy creativity, considered to be the pillar for good designs. Design methods are, however, the product of the scientific approach to design and can be related to and combined into any stage of the design process. As such, in order to facilitate systematic thinking about a design and to develop new design methods there needs to be models of the design process and/or design methodologies. Design theory includes several classes of design process models (Finger and Dixon, 1989):

1. Descriptive models of design processes
2. Prescriptive models for design
3. Computer-based models of design processes
4. Models for Distributed Design Problems
5. Models for Complex Design Problems

These models are described in the following sections. In the section related to models of complex problem solving we have also listed a number of design methods that have been tailored specifically to cope with the complexity at various levels of PPO.

2.1 Descriptive Models of Design Process

Descriptive models of design are *cognitive models* which explain, simulate, or emulate the human designer's underlying mental processes when creating an artefact (Finger and Dixon, 1989). A monumental detailed descriptive model of

the design process was presented by French (1985) that included the following stages or activities:

1. analysis of problem
2. conceptual design
3. embodiment of schemes
4. detailing

The analysis of the problem was regarded as a small yet significant stage of the overall design process. The output of this stage is the problem statement that has three elements:

1. The problem itself.
2. Limitations placed upon the solution, e.g. customers' standards, date of completion, overall cost, etc. all of which can be known as constraints.
3. The design objectives or criterion of excellence.

Table 2.1 presents the problem analysis that contains all the above elements for the design of a hypersonic combat aircraft.

Table 2.1 Need analysis of a hypersonic combat aircraft, from Hollingsworth and Mavris (2000).

Performance/Attribute	***Threshold***	***Desired***
Cruise Speed	Mach 4	Mach 8
Max Speed at sea level	400 Kts	630 Kts
Mission Radius	750 NM	1500 NM
Structure Load Factor	3 G's	5 G's
Takeoff & Landing	8000 ft Runway at sea level	Carrier suitable
Combat Turnaround Time	Less than 6 hours	Less than 2 hours
Alternate Weapon 1	2 JASSMs	8 JASSMs
Alternate Weapon 2	2 AAMs	8 AAMs

The conceptual design phase in French's model (1985) was described as a phase in which the problem statement is taken as an input; and broad solutions in

the form of schemes are generated. In the conceptual phase of design, innovative and striking improvements can be made that require the collaboration of people involved in the production, manufacturing and commercial aspects of the development. The embodiment of schemes stage involves detail clarification of schemes and, in case there is more than one scheme, a final selection amongst them. This phase usually provides feedback to the initial conceptual design phase. In the last phase of detailing a very large number of small but essential points remain to be addressed.

Another descriptive model of design was the model of Cross (2000) consisting of four steps:

1. Exploration: the designer explores the satisfactory solution concepts to a typically ill-defined design problem.
2. Generation: the designer generates alternatives that are like design proposals.
3. Evaluation: the designer evaluates each alternative. This phase, if required, provides feedback to the generation stage to refine the concepts and alternatives.
4. Communication: this stage includes the presentation of the design artefact to the production crew in a form (language or representation scheme) that is understandable to them.

Descriptive models of design processes thus plainly describe the succession of activities that naturally happen in designing. They study the design process in light of how humans create designs (Finger and Dixon, 1989). Descriptive models are solely based on the observations made on the previous design experiences and give no detail of how the design process *ought to* proceed.

2.2 Prescriptive models for design

Prescriptive models for design describe the *necessary and crucial* procedures that the design process *ought to* have (Finger and Dixon, 1989). These models try to

devise ways of thinking that are more systematic and lead to improved (design) process efficiency and effectiveness; however they are often regarded as a particular design methodology rather than design methods (Cross, 2000). These models have tended to suggest a basic structure of analysis-synthesis-evaluation to the design process (Jones 1984). Jones (1984) defined these stages as follows:

1. Analysis: listing all the design requirements and performance specifications.
2. Synthesis: finding possible solutions.
3. Evaluation: evaluating the accuracy of alternative designs in fulfilling the performance requirements for operation, manufacture and sales.

A more detailed prescriptive model was developed by Archer (1984) who identified six types of activities:

1. Programming: establishment of crucial issues and the suggestion on the course of actions.
2. Data collection: collecting, classifying and storing data.
3. Analysis: identifying sub-problems; preparing performance (or design) specifications; reappraising the proposed programme and estimation
4. Development: developing design(s) prototype(s); preparation and execution of validation studies.
5. Communication: preparing the manufacturing documentation

A reasonably comprehensive model known as canonical design process is that offered by Pahl and Beitz (1984). It is based on the following design stages:

1. Clarification of the task: collect information about the requirements to be embodied in the solution and also about the constraints.
2. Conceptual design: establish function structures; search for suitable solution principles; combine into concept variants.

3. Embodiment design: starting from the concept, the designer determines the layout and forms and develops a technical product or system in accordance with technical and economic considerations.
4. Detail design: arrangement, form, dimensions and surface properties of all the individual parts laid down; materials specified; technical and economic feasibility re-checked; all drawings and other production documents produced.

The canonical design process can be regarded as very similar to French's descriptive model only with more details and elaborations. The aim in descriptive models should be to close the gap, as much as possible, between the reality of design (as a mixture of art and science) and the systemic view of design. The model can be further extended (Figure 2.1) to include a manufacturing process selection phase in which the appropriate machines and processes are selected, and a design prototype phase in which concepts are evaluated by soft prototyping using the computer for solid modeling and simulations, as opposed to hard prototyping where concepts that are more refined are fabricated and physically tested It is assumed that the design environment under consideration recognizes the cost, time, functional, and quality benefits of soft prototyping design concepts early in the design process (Reidsema, 2001). Other prescriptive design models include, but are not limited to, an integrative model based on the co-evolution of the problem space and solution space proposed by Cross (2000), parametric model (parameter analysis) of Kroll et al (2001) and Morphological analysis (Allen, 1952).

The integrative model of Cross (2000) is based on the fact that in most design situations it is not possible, or relevant, to attempt to analyse the problem *ab initio* and in abstract isolation from solution concepts. Rather, the designer explores and develops the problem and solution together. As illustrated in Figure 2.1, although there may be some logical progression from problem to sub-problems and from sub-solutions to solution, there is a symmetrical and commutative relationship between problem and solution, and between sub-problems and sub-solutions. This model attempts to capture the essential nature of the design process, in which the understanding of the problem and of the solution

develop together, or co-evolve. There is a constant transfer of the designer's attention backwards and forwards between the problem space (left-hand side of the model) and the solution space (right-hand side of the model). The model recognises that there is an expected pattern of progression in the design process, from a given problem to a proposed solution. There is therefore assumed to be a general anti-clockwise direction of movement in the model (see Figure 2.1), from top left around to top right, but with substantial periods of iterative activity, going to-and-fro between problem and solution, sub-problems and sub-solutions.

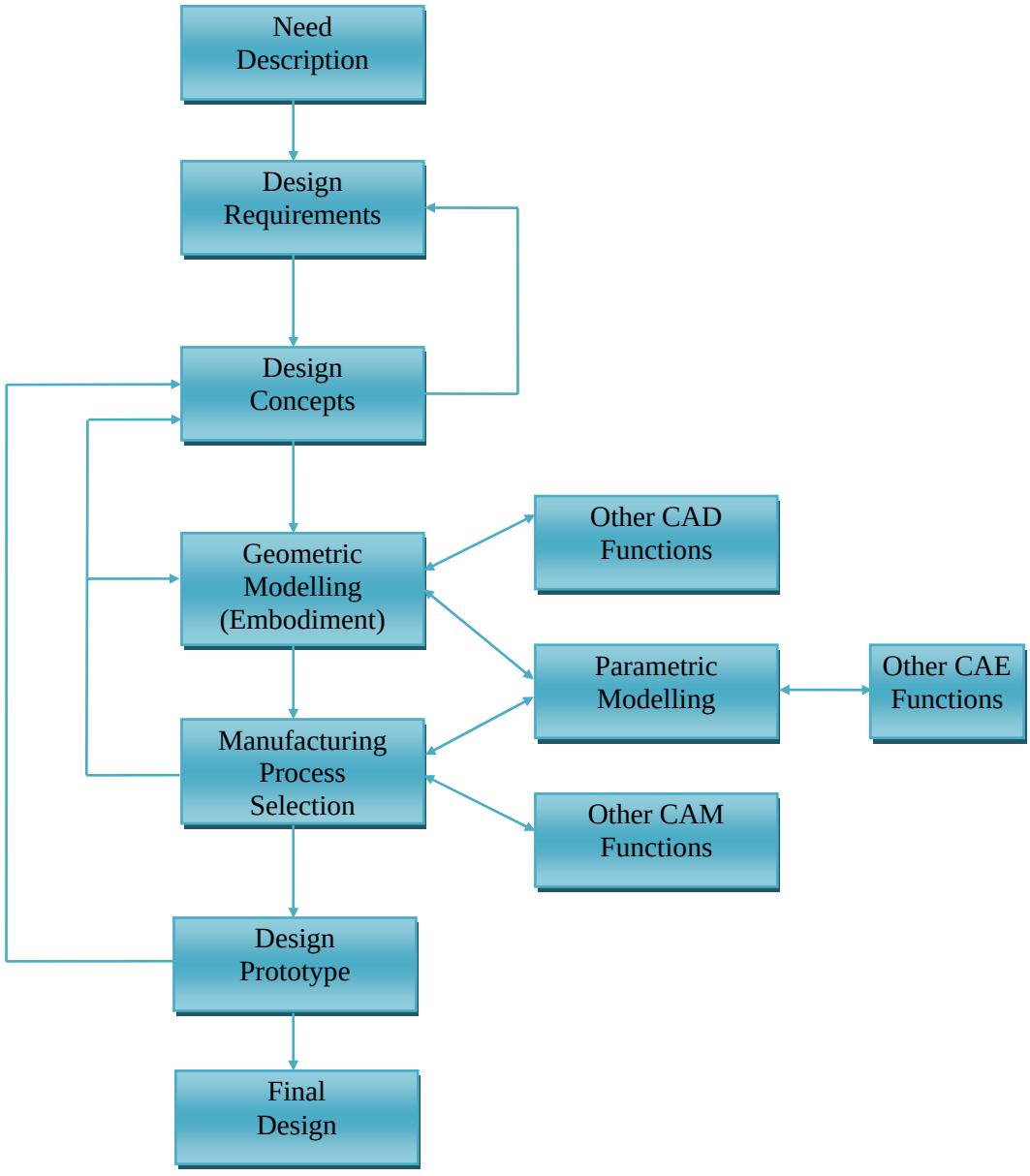


Figure 2.1 Prescriptive Design Process (Reidsema, 2001).

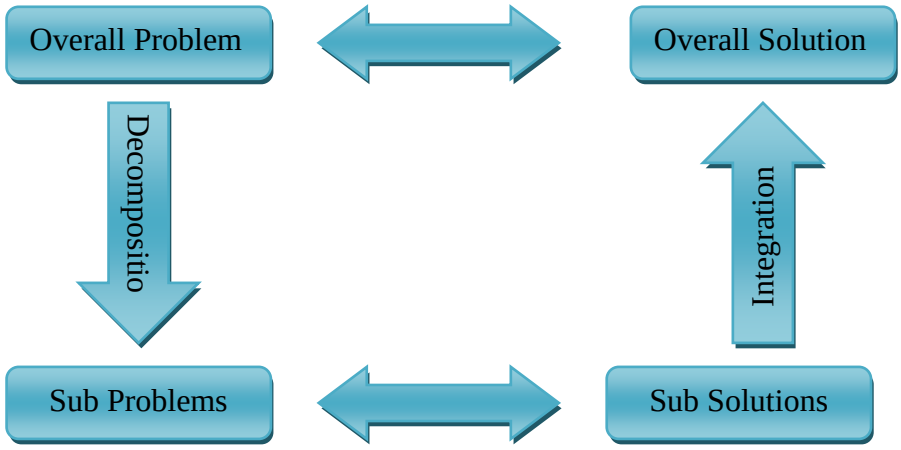


Figure 2.2 The symmetrical relationships of problem/ sub problem / sub solution / solution in design, after Cross (2000).

Kroll et al (2001) presented an important prescriptive design methodology (to the work of this thesis) to improve the creativity of the design process called *parameter analysis*. In this approach parameters are important factors, issues, concepts, or influences that play important roles in the realization of the problem and the development of the product. This methodology has three main elements with significant feedback loop joining the last to the first stage:

1. *Parameter Identification*: recognizing the important parameters. This is the establishment of the problem space. New parameters bring with them new insights and trigger new solutions to the problem and/or stimulate new directions.
2. *Creative Synthesis*: exploring the created space by the various parameters and generating possible solutions. This stage is similar to the configuration and embodiment stage in canonical design process.
3. *Evaluation*: estimating those generated solution.

Figure 2.3 shows that introduction of new parameters bring with themselves their specific solution space in which optimization may be performed. The parametric view of the design process will be discussed in the next two sections in more detail.

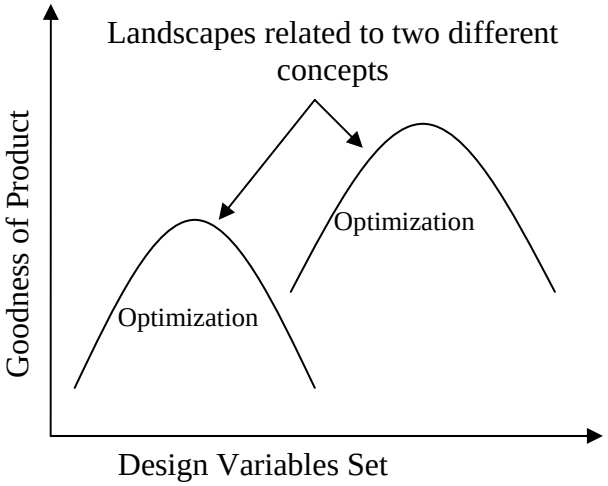


Figure 2.3 New concepts can totally change the design landscape, after Kroll et al (2001).

To choose between several design concepts and configurations (convergence), goodness of design criteria is needed; and this is often determined by the product's performance quality, and/or the overall cost of the final product. Simplicity can be taken as the goodness criteria for concept selection of complex products, the rationale and benefit of which will be elaborated on in Chapter 7.

Morphological analysis is a methodology to generate and select alternatives and has subsequently evolved since its first introduction by Allen (1952). Morphological analysis is based on the following assumptions (Finger and Dixon, 1989):

1. Any complex engineering problem can be divided into a finite number of sub-problems.
2. Each sub-problem can be considered separately and its relations with other sub-problems can be temporarily suspended.
3. All sub-problems and their solutions can be presented in a morphological table.
4. A global solution to any complex engineering problem can be found as a combination of solutions to individual sub-problems.

5. A global solution can be found in an unbiased way through a random generation of combinations of solutions to sub-problems from the morphological table.

A morphological table of the aerodynamic sub-domain of the combat aircraft is presented in Table 2.2. The different functional requirements roughly correspond to sub-domains of aerodynamics, structure, propulsion and control; correspondingly the problem can be decomposed into those categories. Similarly additional morphological tables can be produced for each sub-domain. In Chapter 4 we use the notion of a solution space that can be regarded as a parametric morphological table. Prescriptive models usually require the intervention of human designers, and thus cannot be utilized in computer aided design.

Table 2.2 Aerodynamic sub-domain of the problem space for hypersonic combat aircraft. From Hollingsworth and Mavris (2000).

Type of wing	*small*	*Delta*	*wing tails*	*Wing and tails*	*Swing wing*
Aerofoil	traditional	Diamond	Almond	Biconvex	
Wing location	Tail	Canard	Centre	Multiple	
Body type	wave rider	Partial	Non Waverider		
Body cross section	square	Triangle	Ellipse	Crescent	Other
Body shape	wedge	Cone	Square	Other	
Nose	blunt	Sharp	Spatula	Spike	
Surface Location	Tail	Canard	Centre	Top	Multiple

2.3 Computer-based models of design processes

Computer-based models of design processes are concerned with how computers can design or assist in designing (Finger and Dixon, 1989). A computer-based model expresses a method by which a computer may accomplish a specified task. Computer-based models are generally specific to a well-defined class of design problems such as Conceptual Design, Configuration Design, and Parametric Design (Finger and Dixon, 1989). In each of these design classes, the design automation is achieved by the emulation of a computer based design model that is based, more than anything else, on specific problem representation schemes.

The subject of innovation or creativity often arises in connection with conceptual design (Finger and Dixon, 1989). Innovative conceptual design is a widespread research theme with substantial emphasis on searching past products configurations and then mutating those chosen parts (or the information of the parts) by a genetic algorithm to establish novel configuration (for example see Pramee and Bonham (2000)). CYCLOPS (Criteria Yielding, Consistent Labeling with Optimization and Precedents-Based System) was able to deliver innovative designs by effective searching past product information (Sriram et. al., 1989). The limitation of automation in conceptual design is that it requires specific representation schemes of the product information; and it should be noted that there is no universal representation scheme (i.e. the past product information is not always available in the specific representational scheme). Kurtoglu (2007), for example, used a graph theoretic representation of different configurations and then an automation scheme for conceptual design. In Chapter 6 we address the innovation and creativity in design by utilizing parametric representation of the design problems at the conceptual stage.

In parametric design, usually, the structure or attributes of the artefact are known at the outset of the design process (Zdrahal and Motta, 1996). These include the set of design variables (or inputs) and the set of design objectives (or outputs). Design objectives relate to the functions of the product. The design variable sets, usually, include subsets of sizing variables, shape variables, topologies and process knowledge and manufacturing variables such as process capabilities (Prasad, 1996). It should be noted that the values to be assigned are not always numeric, but may also be a type or class designation, or even an issue e.g., a material choice, a motor type or any other issues (Kroll et al, 2001; Zdrahal and Motta, 1996). Prasad (1996) has defined some of these as follows:

- *Sizing Variables*: these include variables like thicknesses (for thin walled sections) and areas (for solid objects) that can be changed

- *Shape Variables*: These involve changing the configuration points or the geometry of the parts that are represented such as length width, height, coordinates and so on.
- *Topology Variables*: These define parameters that actually determine where material should or should not be removed. As long as the topology change can be represented parametrically in the CAD system, the model can be optimized. Topology optimization allows feature patterns such as how many bolts are needed to hold down a given part, or how many ribs provide a given stiffness.
- *Process Variables*: these involve changing the rules concerning the part's forming or processing needs that have the effect on changing the part's size, shape, topology or functions themselves, cost and lead time.
- *Manufacturing Variables*: these include the process capability indices, and required precisions, manufacturing lead time and cost.

Each variable may be accompanied by a set of constraints. Parametric design problem solving is the process of assigning values to design variables in accordance with the given design requirements, constraints, and optimization criterion (Zdrahal and Motta, 1996). A design task in this view constitutes the determination of a single design variable. Computer models of the design process can be extended to address the problem solving procedure of the large scale design problems.

2.4 Design models for distributed problem solving

For large scale design problems such as aircrafts, cars etc. the design process is carried out by multiple design teams that are more often than not multidisciplinary design teams. In multi-team design, a team refers to a collaboration of design participants that, in a broad sense, can consist of designers, computers, or even algorithms, and in general whosoever that is able to cope with a distributed task as part of the whole design problem (Chen and Li, 2001).

The distributed models generally utilize two different tools: decomposition and abstraction. Decomposition reduces the problem to several sub-problems which may be distributed amongst severally distinct design teams (Figure 2.4). Several decomposition modes are discussed in Chapter 4. Abstraction, on the other hand, decomposes the problem along the axis of time (Figure 2.5). Several abstraction strategies are discussed in Liu et al (2003) and will be presented in the next section.

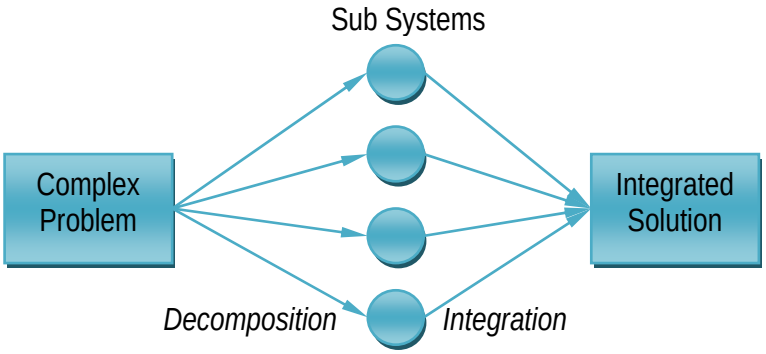


Figure 2.4 Decomposition and integration processes, after Eppinger (1997).

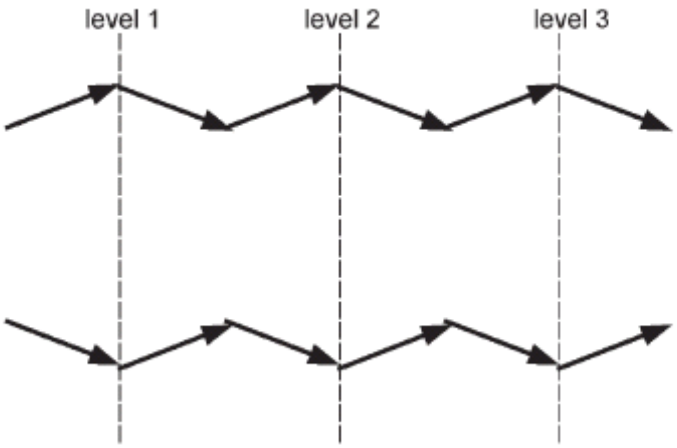


Figure 2.5 A multiple abstraction level design process consisting of several divergence and convergence processes, after Liu et al (2003).

Respectively the integration issue is concerned with the coherence of the product across many abstraction levels (temporally distributed) or sub-problems (distributed to spatially divers design teams). This is of course an organizational

structure type of problem. Coordination strategies are required to handle the coherence problem and the models of distributed design are diverse on how they handle the integration problem. For example Meunier and Dixon (1988) described a computer program for hierarchical distributed problem solving that is based on iterative re-specification. In this model, system and subsystem nodes, called *managers,* each formulate a subsystem design that meets the specifications passed down from a higher manager. Managers meet their specifications by writing specifications for a subsystem and then integrate the sub-solutions into a complete sub-system. The resulting design is then evaluated. If the design is not acceptable, the manager must change the sub-solutions' specifications, hence the term "iterative re-specification" to obtain subsystem solutions that result in a better integrated design. This model did not allow for direct communication among managers that are at the same hierarchical level.

Reidsema (2001) proposed a model for distributed planning that had the elements of both abstraction and decomposition. He described it as a cyclic approach (GDDI cycle), including four stages shown in Figure 2.6.

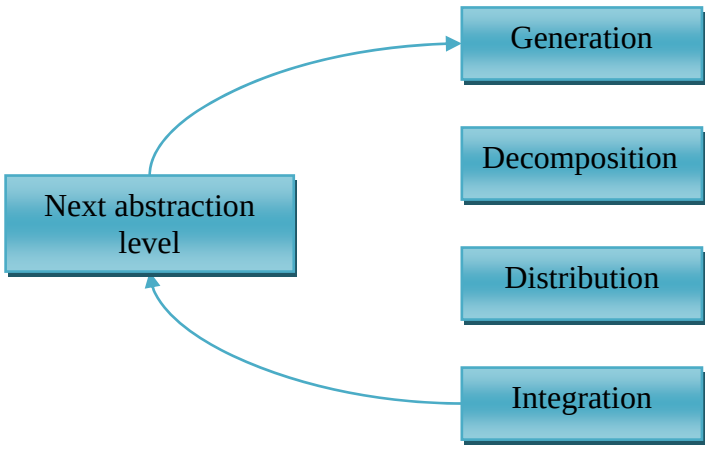


Figure 2.6 GDDI cycle or the distributed model of Reidsema (Reidsema, 2001).

Although this model was proposed in the context of planning for problem solving, it can be regarded as a powerful and generalized problem solving methodology when combined with the parameter analysis of Kroll et al (2001) since it formalizes the utilization of both abstraction and decomposition. The

original planning model consumed the task model of the design process, which is a managerial tool developed in the 1990s for management of the complex problem solving processes and is discussed in the next section. In Reidsema's model the plans comprised a set of tasks. The model can be modified to be applicable to parametric design problem solving (and not only for the planning phase). It would be enough to replace the notion of plan with a set of design variables or in a broader context, parameters that are related to and describe a generated design concept. The design process in this case would be performed in various abstractions or cycles. This is a gradualist approach to design and is adopted in this thesis. However, our approach is simulation based which is a feature of the spiral design process model explained in the next section.

2.5 Design models for complex design problems

Models of complex problem solving are specifically tailored for designing large scale system that embed substantial coupling between their parts, subsystems and components. We believe any model that addresses all or some of the following issues can be regarded as a model of complex design problems:

1. The relationship between global emergent properties of an engineered artefact and local properties of its parts.
2. The control, coordination and cooperation relationships between design teams at the level of NPD organization.
3. The interdependencies of the solutions at different abstraction levels.
4. The integration problem of many interdependent sub-problems at one abstraction level.
5. The management of complex NPD projects prone to desirable cost, time and quality requirements.

This section presents the design methodologies as well as the design methods and design strategies that deal with complexity. The Concurrent Engineering model can be regarded as a model of complex problem solving, since

it considers several upstream issues (or top level issue) at the downstream of the design process. Concurrent Engineering has been devised to rectify the life cycle reliability issues of complex products such as manufacturability, reparability, usability, and assemblability (Reidsema, 2001). This model is derived from, and is a modified version of the canonical problem solving procedure. Concurrent engineering emphasizes the considerable overlap between the tasks of the different stages in the canonical design model. The concurrent engineering strategy as applied to the design process can be viewed as one in which enriched information is continuously or iteratively passed between overlapping phases (Prasad, 1996; Reidsema, 2001). Information builds up within one phase and is released to the succeeding phases as it is needed. As the NPD process progresses, new requirements, constraints and matured information are introduced into the product's design which alter the original design plan (Figure 2.7).

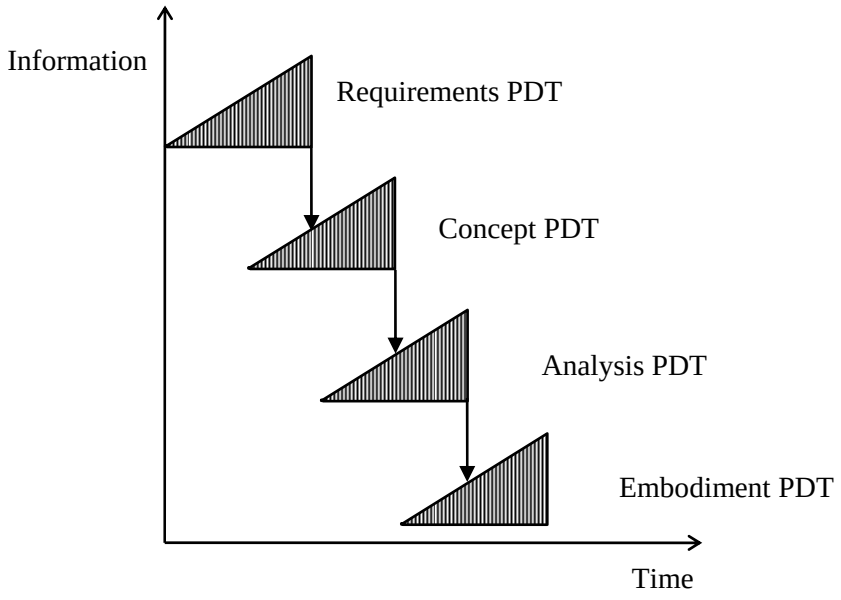


Figure 2.7 Information builds up in concurrent engineering development, after Reidsema (2001).

Another model for complex problem solving is the spiral model of the design process (Figure 2.8). The spiral process has been mainly adopted by software developers to reduce rework, and by that, to lower the development time and cost (Boehm, 1988; Gilb, 1988; McConnell, 1996). This model combines the

features of the prototyping model (simulation based engineering), concurrent engineering model and GDDI cycle distributed model of Reidsema (2001). We will, to a sufficient extent, cover the applications and methods of the simulation in computer aided engineering and design in Chapter 4. The major advantage of the spiral process is multiple cross-phase iterations, which is beneficial for handling difficulties presented by unclear initial product requirements (Unger, 2003). “The spiral process enables brief glimpses into the future (of the design process) by executing each stage with the full expectation of returning back to them later” (Unger, 2003). The information gained from this glimpse can be incorporated into early design activities such as concept generation, requirement specifications and architectures. The look into the future reduces the risks (Unger, 2003). However, Unger (2003) specifies two major disadvantages with the spiral process:

1. The first step of determining objectives, alternatives and constraints is difficult. The difficulty results from the fact that the effectiveness and efficiency of the entire spiral process is sensitive to choices that should be made in the first step.
2. It is difficult to decide on the termination of one stage and the start of another stage.

In Chapter 4 we will address both of these issues first by describing the problem at the initial stage as a seed that will transform or grow into the subsequent stages. Then we introduce a simple termination criterion by considering the abstracting techniques introduced by Liu et al (2003).

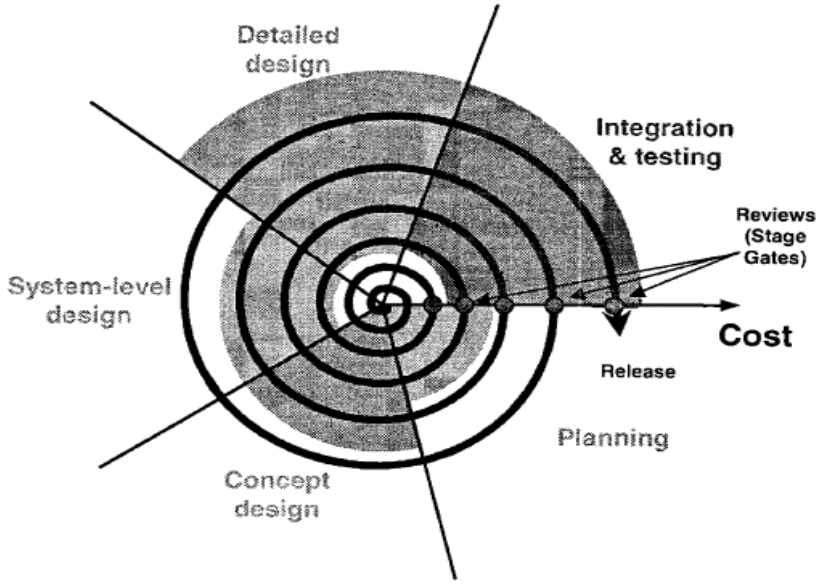


Figure 2.8 The spiral model of design, from Unger (2003).

Another model for complex problem solving is the evolutionary model of the design process (Bar-Yam, 2004; McConnell, 1996). This is based on prototyping and testing multiple versions of the product and competition in between those solutions for higher fitness (Figure 2.9). The evolutionary model of the process is most suitable for design problems that are likely to have emergent properties due to their high complexity.

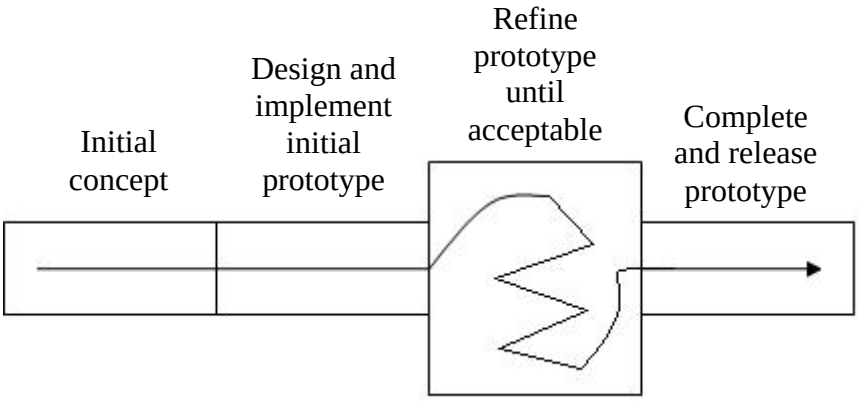


Figure 2.9 The Evolutionary design process model. After Mcconnell (1996).

The Task model was one of the first models that tended to address complexity (Figure 2.10). This model has been developed to address and assess high level estimation of the integration problem and interdependency issues of low level design activities such as parametric design (Kusiak, 1999). According to this model a task or activity has both an information input and an output. The task itself is subject to a control that influences the tasks. A Mechanism is the tool or resource required to perform the task. There are a number of possible models for representing design activities or tasks that are commonly used, such as Program Evaluation and Review Technique, Structured Analysis and Design Technique, and Design Structure Matrices (DSM) (Eppinger et al, 1992; Reidsema, 2001).

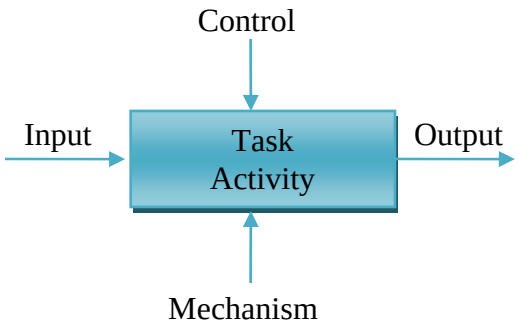


Figure 2.10 Task Based Model represents the low levels design activities as a black box, from Kusiak (1999).

The design structure matrix is a system modelling and knowledge representation tool that is useful in decomposition and integration (Browning, 2001). A DSM shows the relationships and interplay of components of a system as a matrix that has identical row and column labels (Eppinger et al, 1992; Reidsema, 2001; Browning, 2001). DSMs are usually employed in modelling products, processes, and organizational architectures. Browning (2001) presented the following types of DSM:

1. *Parameter-Based DSM (PDSM)**:*** represents the product architecture and is used for modelling low-level relationships between design variables.

2. *Activity-Based DSM:* models the information exchange and dependencies between various tasks of an activity network.
3. *Team-Based or Organizational DSM:* models the organizational structure interns of the information exchanges and interactions between the design players such as design teams.

In general the task based model of design is a managerial tool that allows for identifying activities that have important interconnections to other activities, namely those tasks that consume information output of many other tasks, and also the ones that generate the information input of many other tasks. Obviously the failure of these tasks introduces vulnerability into the entire design tasks network. Braha and Bar-Yam (2006), in an empirical study, showed that complex NPD task networks, in general, are characterized with few of the central tasks on which a specific managerial focus must thus be given. They also showed that these networks show small world properties[5]. These managerial techniques seek to mitigate the complexity of design process by either rescheduling the activity sequences or grouping the more interdependent tasks into similar groups. We discuss this approach in detail in Chapter 5.

An important design methodology was developed by Suh (1988), with the aim of enabling systemic representation and design analysis, a rather advanced description of which can be found in Suh (2005). Axiomatic design is based on the concept of functional requirement, design parameters and their quantitative/qualitative interplays. The design parameters, here, are regarded as means of achieving the functional requirements. The design matrix thus signifies how functional requirements and design parameters are related. A design, in axiomatic design is defined as the interplay between the functional domain (functional requirements) and the physical domain (design parameters) (see Figure 2.11). A design that follows the recommendations of the design axioms is regarded as a *good* design. Originally there were two design axioms, namely Independence axiom and Information axiom but recently Suh (2005) introduced a

[5] A network with small world property is one in which most nodes are not neighbors of one another, but most nodes can be reached from every other by a small number of hops or steps (Source: Wikipedia, online encyclopedia).

third axiom of Complexity that, according to him, demands both axioms of Independence and Information. These axioms are:

1. Independence Axiom: Maintain the independence of functional requirements.
2. Information Axiom: Minimize the information content of design. Information content of the design is whatever information that is required to reduce the uncertainty about achieving the functional requirements.
3. Complexity Axiom: Reduce the complexity of the system.

The main argument in axiomatic design is that the more the functions of the product are satisfied or performed by independent components in the physical domain, then the design process will be simpler to handle and will be more effective at predicting the total cost and overall quality performance. However, another trend in design process models of complex products is to take a different viewpoint and suggest the exploitation of emergence and more complexity (rather than less complexity) in the functions of the product and thus at the conceptual design level.

Weber and Condoor (1998) recommended exploiting synergetic effects of combining design solutions from the morphological table. According to them synergy is achieved when one component (or subsystem) can meet the criteria of more than one functional requirements. They explained a top-down approach based on abstraction:

1. Identify independent primary functions.
2. Create solutions for primary functions.
3. Create primary morphological matrix.
4. Choose a compatible synergistic solution.
5. Identify lower-level functions.
6. Create lower-level solutions.
7. Create lower-level morphological matrices.
8. Choose a compatible synergistic solution.

9. Evaluation: go to step 5 if more detail is required.

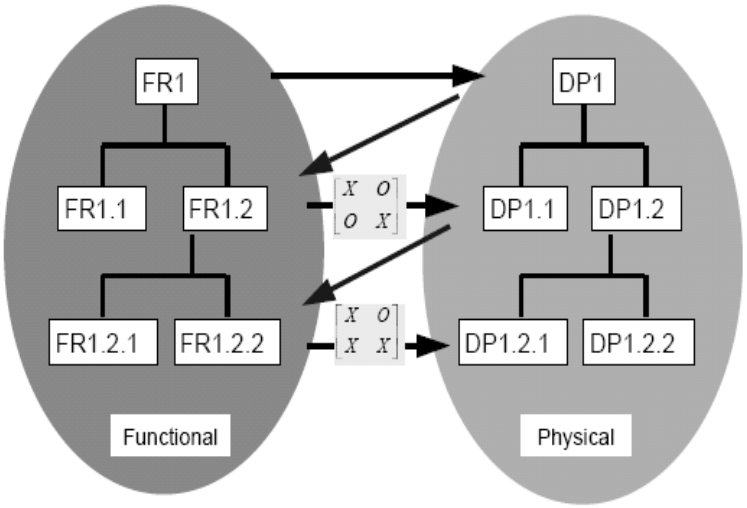


Figure 2.11 Design matrix in axiomatic design shows the coupling of the physical domain and functional domain, from Lee (2003).

This approach however fails to address the synergy between different abstraction levels. One possible solution to this problem was proposed by Liu et al (2003). They posed that for complex problems an ideal approach to abstraction would be to carefully specify the number of possible solutions to be considered at the divergent stage of each abstraction level in order to have a global trend of divergence-convergence. Figure 2.12 demonstrates all the possible scenarios in the global divergence-convergence patterns that Liu et al (2003) recognized.

Figure 2.12(a) demonstrates an approach that first only carries out the synthesis activities at each solution abstraction level until all possible solutions are generated, and then evaluates and selects these concepts. Divergence of all abstraction levels takes place afterwards in the final stage. This method provides the opportunity of considering synergetic solutions at different abstraction levels, but since the solution space would be too large, the search for those solutions may be difficult. Figure 2.12(b) demonstrates an alternative approach in which we carry out divergent and convergent activities at each level of the solution abstraction. This should allow a reasonable number of concepts to be generated at

each solution level (divergent step), immediately followed by a screening of these concepts (convergent step). By means of these multiple divergent and convergent steps, the management of the solution space is possible. The challenge here is that solutions represented at an abstract level (e.g. functional level) can be hard for designers to understand. It is a question of how to screen an abstract solution space. Figure 2.12(c) shows the classical design process approach to multiple levels of abstraction posed by Pugh (1991) and Cross (2000).They stated the necessity of the global trend of the solution space size to be towards convergence. This means that the number of concepts must gradually be decreased and only one or few solutions must be left at the end of the design stage. Figure 2.12(d) shows multiple solution abstraction levels with the global trend towards divergence.

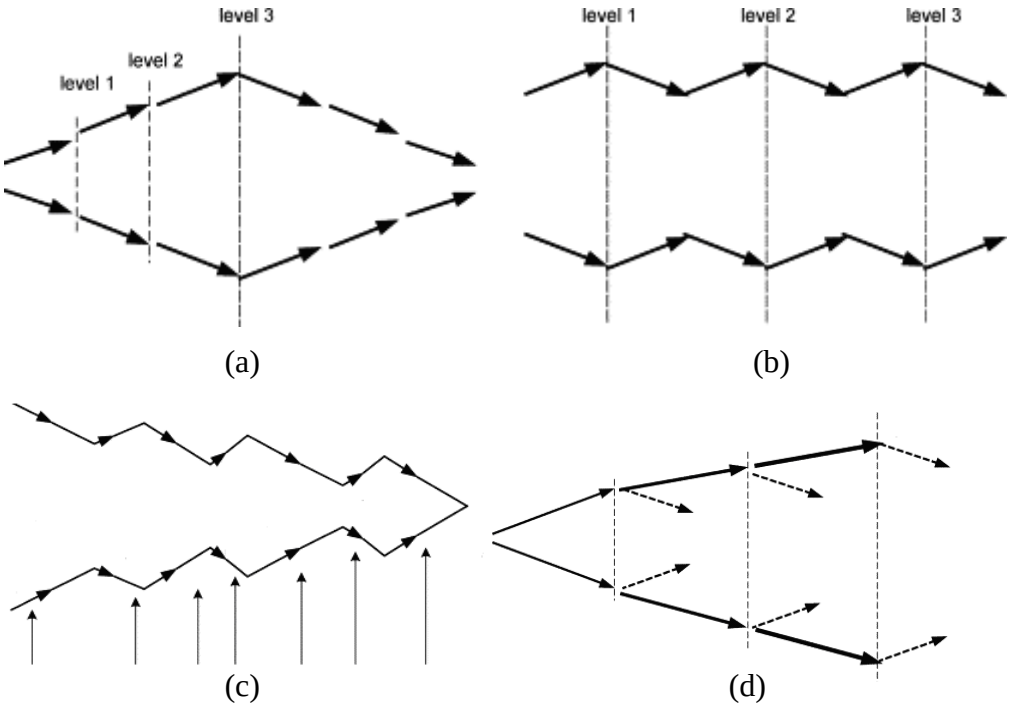


Figure 2.12 Shows multiple divergent followed by multiple convergent processes (a), multiple consecutive divergent-convergent processes (b), with a global trend towards smaller solution space (c), with global trend towards larger solution space (e), after Liu et al (2003).

The 'ideal' approach of concept generation according to Liu et al (2003) (Figure 2.13) should follow multiple divergence—convergence in order to gradually increase the number of solutions for the concepts generation, followed

by a divergent and convergent tendency to detail these concepts with an overall decrease in the solution number. This way the design process would benefit both from synergetic solutions and also a manageable search in the solution space.

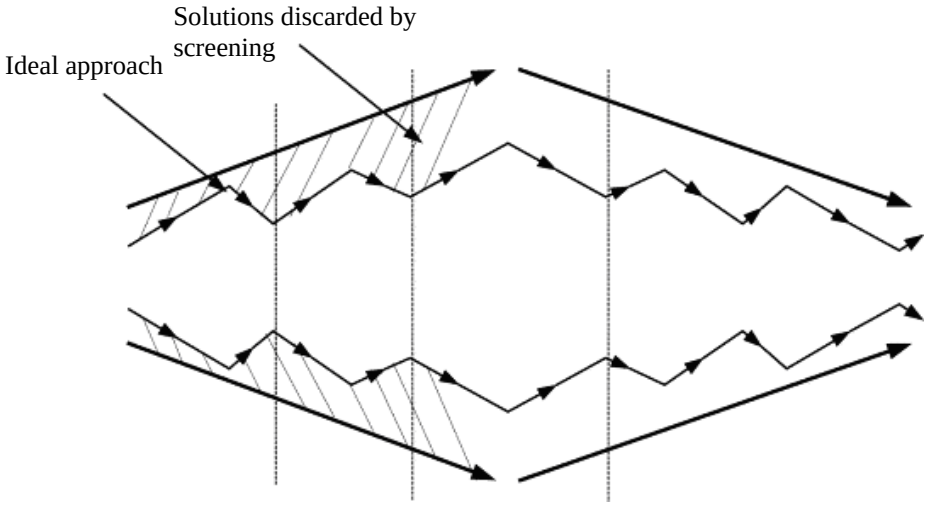


Figure 2.13 The ideal design abstraction strategy of Liu et al (2003).

In a seminal paper, Alstyne and Logan (2007) called for the redesigning of the design process in a way that can harness emergence:

> Only through emergence new innovative products emerge which can revitalize and also survive in the global economical market. All the great innovations of the history harnessed emergence: tool making of the early man; the internet; Gutenberg's printing press etc. Nature's products have emergent properties and functions that are performed by the cooperation of millions of micro-organisms and several parts.

Alstyne and Logan (2007) posed that a homeostatic relationship between design and emergence is a required condition for innovation. They called for employing emergence and self-organising processes as bottom up and massively

iterative processes. Anderson (2006) described how bottom up[6] strategies at the NPD organization level can harness emergence in the product and in the process. Bar-Yam (2004) called for forsaking the planning based method (that renders the process utterly top down) and allowing for radical innovation to emerge. Bar-Yam (2004) believed that only through employment of radical innovation, could the integration problem of complex systems could be addressed. This thesis takes a moderate position, which is to use low level design problem knowledge to configure and control high level organizational structure (which can be regarded as planning). This is discussed in Chapters 4 and 7.

We present a model of design process for complex products that is, distributed, evolutionary and has elements of the spiral process. It is based on the parametric representation of the design at various abstraction levels and uses the low level parametric knowledge of the problem to plan the development organization such that emergence of innovation becomes a possibility. The presented model benefits extensively from complexity measures at three levels of PPO.

[6] Top down design asserts the role of hierarchical command and control, in contrast bottom up design asserts the pivotal role of low level design agents in shaping the overall and emergent properties of the design artefact. In essence bottom up view calls for flat organizational structures.

3 All about Complexity

It is important to better understand why complex systems, be they complex organizations, complex design processes, design projects, and complex products, possess high potential for failure. In order to attain this goal we first need to define the notion of a *complex phenomenon*.

A phenomenon is any observable occurrence[7]. Observation is the activity of sensing and assimilating the knowledge of a phenomenon, or the recording of its data by utilizing instruments. The set of techniques for investigation of a phenomenon is referred to as the scientific method. These techniques entail data collection by means of experimentation and observation, the formulation of hypotheses, and testing. A complex phenomenon is one that initially may not be observed straightforwardly and could be confused with pure randomness. Furthermore, a complex phenomenon may not be hypothesized by simple means; and even when the complex phenomenon is hypothesized, it may not be tested without difficulty. In this Chapter we show that measures of complexity serve well in demonstrating and explaining these statements. As the scientific method is based on measurement, a science of complexity without a measure of complexity would not be valid (Marczyk and Deshpande, 2006).

3.1 Complexity Measure

The most commonly agreed definition of complexity as a measure is the size of the minimal description of a phenomenon (a system, or an object) when expressed in a chosen vocabulary (Crutchfield, 1994). This is also known as algorithmic complexity, Kolmogorov-Chaitin complexity or deterministic complexity (Crutchfield, 1994). It should be obvious that, this measure of complexity would increase proportionally with the observed randomness in the phenomenon[8]

[7] Definitions in this section are taken from Wikipedia the free Encyclopaedia.
[8] Randomness is the absence of pattern, relation, meaning, or relevance.

(Figure 3.1). However, what is missing in the definition of deterministic complexity is the role of the observer and this is exactly why the measure is referred to as “deterministic”: it is a subjective measure. Because of this it is generally accepted that deterministic complexity is not computable. For example, Crutchfield (1994) explained that:

> Kolmogorov-Chaitin (deterministic) complexity requires accounting for all of the bits, including the random ones and is dominated by the production of randomness and so obscures important kinds of structure.

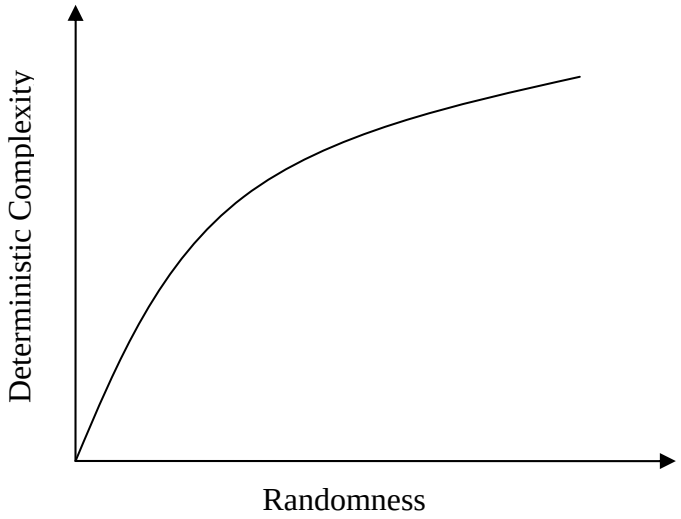


Figure 3.1 Deterministic Complexity increases with randomness, from (Crutchfield, 1994).

Science now agrees that the observer sees through his/her cognitive models, descriptions of objective reality and conceptualizations (Minati and Pessa, 2006). The observer is an integral part of the phenomenon with an active role and not a passive one. This active role is reflected in the limitations of the observation. In general, there are three intrinsic limitations with any observation (Minati and Pessa, 2006; Kuras, 2007):

1. Inherent lack of subjective knowledge in the observer’s mind; thus there is a limitation in the scope of the observation.

2. Inherent limitation of the resolution of the observation. How much of the phenomenon have been observed is dependent on the resolution of the measurement and detection devices.
3. Limitation in computational resources, such as memory and computational power. These lead to and reflect in a limitation of the scope of the observation.

Therefore the amount of regularities, patterns, etc. that are observed depend by and large on the observer.

In contrast to deterministic complexity, stands *statistical complexity* which discounts the computational effort in simulating what is seen as random (Crutchfield, 1994): “statistical complexity is the minimum amount of information required to optimally hypothesize the phenomenon”. Thus statistical complexity is not a measure of randomness and is a measure of structure above and beyond that describable as ideal randomness (Crutchfield, 1994). The fundamental property of statistical complexity is that it is both zero for an ideal random phenomenon and also for an ideal ordered phenomenon (Figure 3.2). In this regard a complex phenomenon to the observer with limitations in resolution and computation resources is one that fits between order and chaos (Duin and Pekalska, 2006).

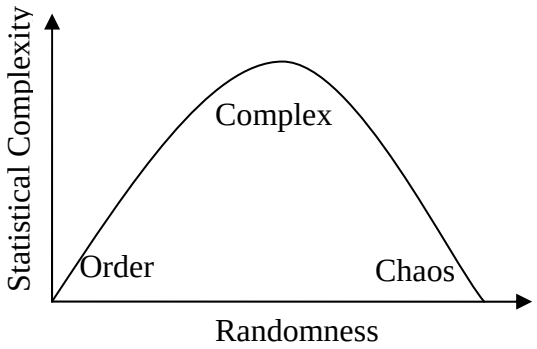


Figure 3.2 Statistical complexity measures the complexity with reference to the observation characteristics (resolution and scope), after (Crutchfield, 1994).

Figure 3.3 presents three patterns that clearly demonstrate that statistical complexity is an intuitive measure of structure. The middle pattern (b) has a relatively higher complex structure when compared to the other two patterns. Pattern (a) is very ordered and is not complex. While the far right one, pattern (c) seems to be structure-less, it is not complex especially when it is thought of as being produced by pseudorandom number generators (Grassberger, 1989). In this case the distinction about the complexity levels of three patterns is made relative to the resolution and accuracy of the observation and thus the inferred complexity is statistical.

(a) (b) (c)

Figure 3.3 Three patterns, after Grassberger (1989).

However, deterministic complexity can identify structure existing even in pattern 3c that seems purely random. Pattern 3c truly has a high deterministic complexity relative to the other two 3a and 3b. The proof is in Figure 3.4. These patterns in fact belong to the same pattern but are at different scales. An immediate consideration is that scale or level of description can change both the statistical and deterministic complexity. This means that hierarchical systems pose different structures and thus different properties at different scales. These properties are emergent and do not exist in the entities of the lower levels (Minati and Pessa, 2006). A simple example of an emergent property is the temperature of gas. At the atomic level, the temperature of a single atom is meaningless and irrelevant. The relation between complexity and emergence is discussed in Section 3.2.

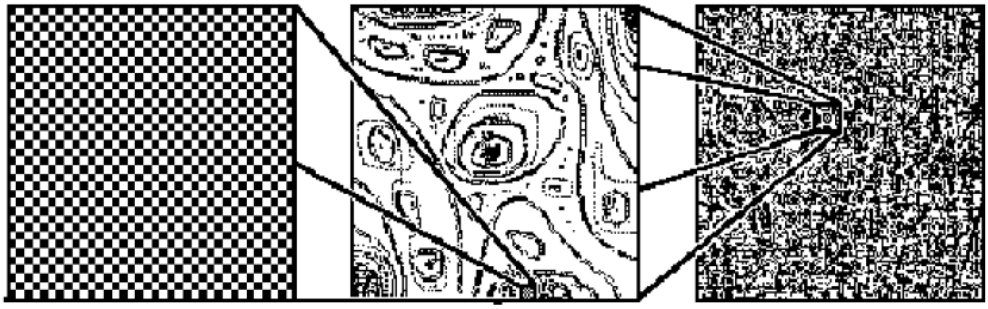


Figure 3.4 The three patterns belong to the same pictures seen at three different scales. After Grassberger (1989).

It should be indicated that a necessary departure from a deterministic approach to stochastic approach (in modelling and hypothesizing) is essential only when the statistical complexity is not far away from deterministic complexity (Figure 3.5). The difference between statistical complexity and deterministic complexity tends to become higher after the statistical complexity has peaked. When the difference is high it is irrelevant to hypothesize, test or even fully observe the complex phenomenon. As mentioned before this is so since the observer has limitations accordingly in scope, computational resources and resolution.

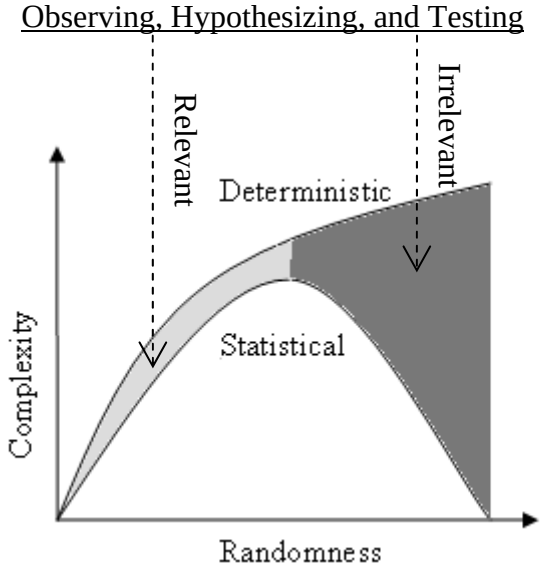


Figure 3.5 High complexity makes the endeavours of the scientist, in hypothesizing, hypothesis testing and even observing, irrelevant activities.

3.2 Complexity and Emergence

Edmonds (1999) defines complexity as:

> ... that property of a model which makes it difficult to formulate its overall behaviour in a given language, even when given reasonably complete information about its atomic components and their inter-relations.

This definition couples complexity with emergence. Complexity as structure gives birth to emergent properties. Emergent properties of complex systems are hard to predict. Although we may not be able to exactly describe the emergent property; we can argue about the potential existence of emergent properties. Therefore a comprehensive definition of complexity would be *the intensity of emergence*. This is a fundamental notion in this thesis and is in accordance with the notion introduced by Marczyk and Deshpande (2006) that complexity is a *potential* (for creating top level properties and overall functionalities). For example a car is relatively more complex than a bike and it has also more functionalities. So is a human community (more complex) relative to an ant community and has relatively more functionalities. Complexity allows the potential for emergence be it desirable emergent properties (functionality) or catastrophic ones (surprise failure).

It should now be clear why complex systems possess potential for catastrophic failure: because complexity can be mistakenly dismissed as noise (therefore not be observed), and emergent properties of complex systems cannot be modelled, or even tested (Bar-Yam, 2004). The failure examples in Chapter 1 all had their own specific reasons, however, there is common agreement that if anything can go wrong it eventually will. This is known as Murphy's Law, a lesson from history.[9]

[9] From Wikipedia online Free Encyclopaedia.

3.3 Graph Theoretic Measure of Complexity

We defined what a statistical measure of complexity must represent but did not give a detailed account of how to measure it. Statistical complexity is a statistic calculated from a data set. The data set represents the observations made about a phenomenon or a system. The first attempt to measure statistical complexity was reported in Crutchfield and Young (1989), who initially proposed the concept as well. In 1997, Edmonds (1997) presented a database of complexity measures containing 386 entries, which according to Shalizi (2001), was not comprehensive. We do not attempt to review these measures; however an extensive literature review of complexity measures in engineering design is presented in 3.4. The central idea in measuring statistical complexity though, is to use entropy based mutual information as a measure of dependency between two variables (Edmonds, 1999). The main property of mutual information is the ability to capture both linear and non-linear relationships, making it more attractive than the covariance based correlation coefficient (Boschetti et al 2005).

Equation (1) shows the entropy based mutual information between two random variables *X* and *Y*. This requires estimation of the probability distribution of every random variable ($P_X(x)$, $P_Y(y)$) in the data set and also the mutual probability distribution of all pairs of random variables ($P_{X,Y}(x,y)$).

$$I(X,Y) = \iint P_{X,Y}(x,y) \times \log \frac{P_{X,Y}(x,y)}{P_X(x) \times P_Y(y)} dxdy \qquad (1)$$

For example, if two variables have a circular relationship (Figure 3.6), then the covariance and thus the linear correlation coefficient in between them is *zero*, whereas the entropy based information exchange in between the variables *x* and *y* is 3.18[10]. Since the mutual information is a number between *zero* and *infinity*, it can be normalized to be a value between *zero* and *one*. One such normalized correlation coefficient based on the mutual information exchange is

[10] The Matlab™ codes that produce the data set presented in Figure 3.6 as well as the mutual information exchange are included in Section A.3.

the global correlation coefficient, a description of which can be found in Soofi (1997), Darbellay (1998) and Dionisio et al (2007).

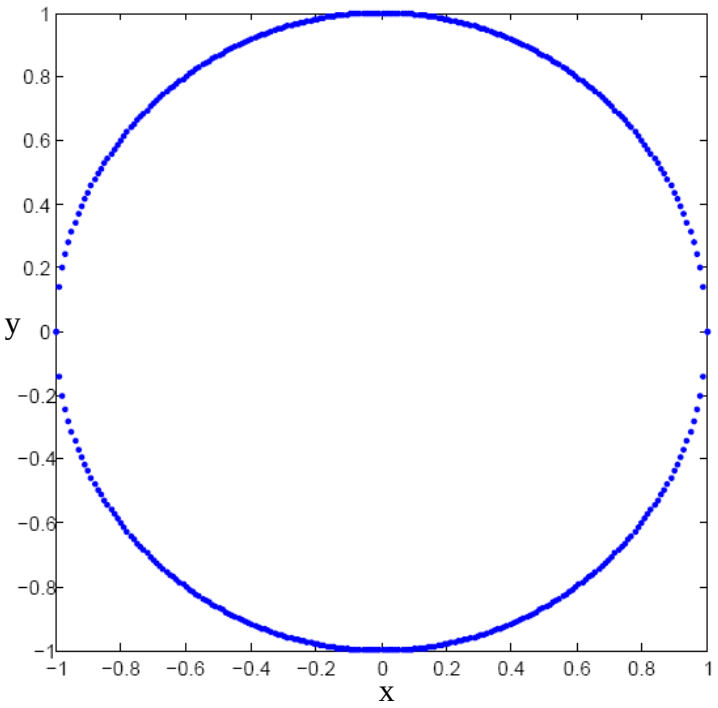


Figure 3.6 The entropy based mutual information captures linear as well as nonlinear relationships.

After measuring all the mutual information of pair of variables in a data set, a graph theoretic measure of complexity can be adopted to capture the amount of structure in the data set (Boschetti et al 2005). But in order to do that the system must be represented in graphical format. A graph *G* can be characterized by its vertex set, *V = {1,…,n}*, and the edge set *E*. The total number of nodes in *G* is denoted by |*G*| and referred to as the order of *G*. The number of edges in *G* is the size of *G* and denoted by *E(G)*. *G (n, m)* is a graph of order *n* and size *m*. Associated with every graph *G* is its adjacency matrix, A_G, which entries ($a_{i,j}$) are defined as:

$$a_{i,j} = \begin{cases} w_{i,j} & (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \qquad (2)$$

For un-weighted graphs all $w_{i,j}=1$, and in undirected graphs for all $(i,j) \in E$, $a_{i,j} = a_{j,i}$. Another matrix that can be associated with graphs is the Laplacian matrix. A Laplacian matrix is defined as $L(G) = D - A$ where D is $n \times n$ diagonal matrix with entries $D_{i,i} = d_i$ and d_i is the degree of the vertex i defined as the total number of edges (or sum of the weights of the edges) that touch the vertex. Hence, every system can be represented elegantly by graphical means. Each variable in the data set can be symbolized as a vertex and each relationship in between the variables is an edge with the given weight (which for example can be determined by entropy based correlation).

With regards to the complexity of a system and its structure being represented with a graph, there are different perspectives amongst system researchers as to what represents a system's complexity. The general belief is that the complexity can be fully represented by size, coupling and cyclic interactions. These are discussed next and after that complexity measures in engineering design literature is reviewed.

3.3.1 Size

There is clearly a sense in which people use "complexity" to indicate the number of parts. The size of a system is the minimum number of variables that the system can be described with (order of its graphical representation)[11]. The notion of minimum size overcomes some of the inadequacies of mere size as a complexity measure (Edmond, 1999): it avoids the possibility of needless length. Size (or order) has been used in the literature as a measure of complexity in applications that include (Edmond, 1999): the social organisation and community size (Carneiro, 1987); the minimum number of gates in a circuit (Lazarev, 1992); the cyclical behaviour of systems (Walker, 1971); self-replicating sequences (Banzhaf, 1994); rule-based systems (O'Neal and Edwards, 1994); neural networks and cellular automata (Gorodkin et al, 1993); and grammatical development (Kemper, 1995).

[11] Systems of high order are usually known as large scale systems.

3.3.2 Coupling

Coupling (or connectivity) is the sum of the densities of dependencies between the system's variables (size of the graphical representation). The coupling of a system is a strong indicator of its decomposability: it is difficult if not impossible to decompose a system with densely interconnected elements/components without changing the overall characteristics of the system (Edmonds, 1999). Applications of coupling as complexity measure include (Edmonds, 1999): the reliability of circuits (Winograd, 1963), the stability of random linear systems of equations (Ashby and Gardner, 1970), stability in computational communities (Kindlmann, 1984), stability in ecosystems (Casti, 1977; Lakshmanan et al, 1991; Pimm, 1984), the diversity of ecosystems (Margalef, 1984), the structure of memory (Kroll and Klimesch, 1992), logical and computational properties of bounded graphs (Meinel, 1990), competition in networks (Reggiani, Nijkamp, 1995), random digraphs (Seeley and Ronald, 1992), chemical reaction mechanisms (Zeigarnik, and Temkin, 1996).

3.3.3 Cycles

The number of Independent Cycles is a basic graph measure sometimes referred to as cyclomatic complexity and is the number of independent cycles or loops[12] in a graph (McCabe, 1976). As indicated in Chapter 1, complex systems are characterized by circular causality. Thus a graph theoretic measure of complexity must point to circularity of dependencies between the system's variables. In general there is no direct relation between the order (number of vertices) and the cyclomatic complexity: the number of vertices will limit the cyclomatic complexity but this effect is only significant with very few vertices as the number of possible edges goes up exponentially with the number of vertices (Temperly, 1981). McCabe (1976) uses this as a measure of program complexity, in particular to calculate the number of different logical paths through a program to gauge how many tests it might need. Other applications include: complexity of simulation

[12] Number of independent cycles is easily calculated by the formula $c(G) = m - n + p$ where m is graph size, n is graph order, and p is the number of independent components determined by multiplicity of zero in eigenvalue spectrum of Laplacian matrix.

models (Schruben, and Ycesan, 1993), and the difficulty of software maintenance (Bechir and Kaminska, 1995; Curtis et al, 1979; Hops and Sherif, 1995).

3.4 Complexity Measures in Engineering Design

In engineering design, there are three elements that may be externally represented by a designer and for which complexity can be measured: the design problem, the design process, and the design artefact (Summers and Shah, 2003). Figure 3.7 shows that by utilizing the process, a problem is transformed into an artefact.

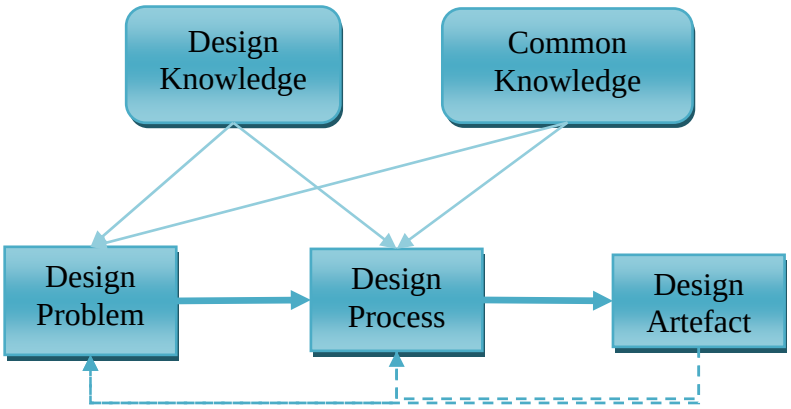


Figure 3.7 Relations between Problem, Process, Artifact, after Summers and Shah (2003).

The design problem is a statement of the requirements, needs, functions, or objectives of design. The design problem is a structured representation of the specific question or situation that must be considered, answered, or solved by the designer (Summers and Shah, 2003). This thesis assumes that the design problem is a collection of variables (numeric, geometric, functional, configurational, etc.). Problems may include the evaluation criteria for the generated solutions. In this light, a design problem consists of a collection of design goals, independent

design variables, measures of goodness, and design relations[13] (Summers and Shah, 2003).

The design process is the method that is used in guiding the problem solving. The design process includes the domain knowledge available to the designer (Summers and Shah, 2003). This domain knowledge may be explicitly represented in rules, design procedures, design manuals, previous solutions, etc. The design process includes both facts and how to apply those facts. Thus, the design process has the following elements (Summers and Shah, 2003):

1. Design problem.
2. The background knowledge is used to modify either the design problem or the design artifact by changing values to existing variables through satisfaction of the existing design relations or by introducing new variables and relations as needed.
3. The design tasks that are the collection of sub-processes performed during the design process.

The design artifact is a representation of the envisioned physical solution to the design problem through the realization of the design variables such that the design constraints are satisfied (Summers and Shah, 2003). The design artifact is the result of the design process when applied to the design problem. The dependent design variables may take the form of parameters or geometric/topologic entities.

It is important to notice that metrics in general can be either result or predictor oriented (Bashir and Thomson, 1999.a). A result metric is an observed characteristic of a completed system such as development time and design effort. A predictor metric has a strong correlation to some further result, such as product complexity, design difficulty, etc., as they relate to design effort or duration.

[13] Relations are the how the design variables, and function variables effect each other.

Complexity measures that apply to the design problem are predictors whereas measures that apply to the design artifact are result. In the proceeding subsections a literature review of various types of complexity measures for each of the three design elements and how/why they have been regarded *useful* is presented.

3.4.1 Complexity Measures for Design Problems

Within the problem domain, measuring complexity has been regarded useful because it can give a quantitative estimation of problem solving difficulty, the required problem solving effort or design effort, design lead time, cost and risk[14] (Bashir and Thomson, 1999.a). Measuring the complexity of a design problem allows for planning by using the results (design process or design artifact) from previous comparably complex design problems to predict necessary resources, time, or commitment required for the new design problem (Summers and Shah, 2003). Thus it is important to have measures of problem complexity.

One primitive problem complexity measure was introduced by Griffin (1993) and Kannapan (1995) as the number of functions included in the functional requirements and to be delivered to the customer. Bashir and Thomson (1999.b) argued that this metric is not realistic since it is insensitive to the complexity of each function and the relative difficulty of developing functions which are more complex. They proposed a simple metric based on the concept of functional decomposition which assumed that product complexity depends on the number of functions and the depth of their functional trees (hierarchies). Their measure was given as:

$$PC = \sum_{j=1}^{l} F_j \times j \qquad (3)$$

Where *Fj* is the number of functions at level *j* and *l is* the number of levels in the functional decomposition hierarchy. These measures however only indicate the

[14] Design risk is the probability of not satisfying the functional requirements at the end of the design cycle.

size of the problem. Dierneder and Scheidl (2001) further advanced the trend of the functional tree based measures and introduced a complexity measure for the design problems that gave strong emphases to the coupling between the functions. The complexity value was the sum of all the functional coupling information contents up to a certain level in the functional decomposition hierarchy. The function based measures give estimates of the design difficulty.

Dierneder and Scheidl (2001) also introduced two other complexity measures: Technical Product Complexity, and the Reliability Product Complexity. These two measures reflected correspondingly the coupling degrees of design parameters to the functional requirements and the amount of uncertainty in achieving the functional requirements by the given set of design parameters. Their approach was based on the representation format of axiomatic design of which three major flaws can be found. First, they do not clarify how the dependencies may be determined. They are assumed to be known to the designer. Second, all the correlations including those between the design parameters, and functional requirements are calculated based on the linear correlation coefficient which cannot capture nonlinear dependence. Third, they did not account for cyclic dependencies that are the root cause of emergence properties.

Complexity as the amount of uncertainty[15] in achieving the functional requirements has been introduced earlier by Suh (2001). Consider P_i as the probability that the i_{th} functional requirement would be satisfied Then Suh (1989) defined Real Complexity as:

$$C_R = \sum_{1}^{n} \log_2 \frac{1}{P_i} \tag{4}$$

Where n is the total number of functional requirements. This complexity measure is redundant and unnecessary; because it is not clear what a measure of complexity as a function of uncertainty has to offer that an estimation of uncertainty by itself could not (Crutchfield and Young, 1898). It would appear

[15] This is more robustly known as the risk involved in a design process.

that Suh's definition of complexity is based on functional requirements uncertainty being solely due to variability and noise particularly those due introduced in manufacturing. It is the argument of this thesis that the major source of risk (in form of surprise) as an emergent property is complexity itself, not the other way around.

Suh (2001) also presented the imaginary complexity as a degree of uncertainty based upon the designer's lack of understanding. This view of complexity attempts to explain why two designers may have different levels of difficulty handling the same problem. This approach includes the designer while measuring the "system" (design problem and designer). This uncertainty may be measured as the probability of the designer "stumbling" upon a solution.

Braha and Maimon (1998) presented two definitions of complexity in engineering design: functional and structural. Their functional complexity was the same as Suh's real complexity. They, however, defined the structural complexity of design as the information content of the minimal representation of the artifact. In this definition information is whatever that is represented by the designer. The difference between the functional and structural complexity according to them is reflected in the difference between design and design efficiency effectiveness. Braha and Maimon (1998) supposed a design artifact as a collection of operands (entities) and operators (relationships). Given a design with *n* operands and operators $\{X_1, X_2, X_n\}$, each of which with a distinct probability distribution, they defined the structural complexity as the entropy of the joint probability distribution of all X_i s:

$$H(X_1, X_2,\ldots, X_n) = \sum P(X_1, X_2,\ldots, X_n) \times \log P(X_1, X_2,\ldots, X_n) \qquad (5)$$

According to Braha and Maimon (1998), a structural definition of complexity has several appealing properties:

1. Simplicity in evaluation of design complexity

2. Simplicity in knowledge exchange between different computer aided design systems since two parts of information can be added together.

And also some limitations:

1. It hardly finds relevance to satisfying design objectives.
2. It is dependent on the method of information acquisition, which means that the information may not be interpretable after the information acquisition process is stopped.
3. It cannot explain the designers underlying psychological and reasoning patterns.

Structural complexity is the engineering version of the Kolomogorv-Sinai entropy (or deterministic complexity) that was originally introduced in the context of dynamical systems theory. This notion of complexity is incomputable in exactly the same way as deterministic complexity (See Crutchfield (1994) for proof). Roughly speaking we can argue that the estimation of joint probability distributions of all design entities by rationally bounded[16] design agents (observers) is prone to error to the degree that the resulting complexity measure would not reflect the collective inter-dependencies (or complexity) of the design problem. We must, therefore, restrict ourselves to estimation of pair wise joint probability distributions of design variables (Fraser, 1989). Braha and Maimon (1998) solved Equation (5) only for when all the design entities were independent (with no interrelation) which is far too simplistic and renders the complexity measure purely as the function of design information content size.

[16] The concept of bounded rationality accounts for the fact that perfectly rational decisions are often not feasible in practice, due to the finite computational resources available for decision making. (Source: Wikipedia)

El-Haik and Yang (1999) tried to incorporate the measure of structure as a component of complexity into the complexity measure. They posed that given:

$$\{FR\} = [A]\ \{DP\} \tag{6}$$

Then complexity as the uncertainty in achieving the functional requirements (Suh's definition) has three components:

1. Variability which is the uncertainty in design parameters measured by their total entropy. Given each of *n* design parameters follow a distinct probability distribution P_i the entropy would be:

$$h = \sum_{1}^{n} P_i \log P \tag{7}$$

2. Vulnerability of the design that was the determinant of domain mapping matrix, which hints at the overall sensitivity of functional requirements to design variables. This is perhaps the closest the literature has got to the approach of this thesis.
3. Correlation or coupling between the design parameters.

El-Haik and Yang's (1999) method takes into consideration the size, coupling and possibly the circularity of the interdependencies between design variables (obviously, in the latter case, without them being aware of it). However, again, it was based on linear correlation coefficients. This measure could be used to compare different design solutions and as such was a measure of artifact complexity as well. In general, axiomatic complexity suffers from applicability issues and that limits the usefulness of the measures that are developed on the basis of this theory.

Summers and Shah (2003) proposed that complexity measures must have three components of solvability, size, and coupling. They presented the following definitions:

1. Solvability is whether the design artifact may be predicted to satisfy the design problem.
2. Size of several elemental counts including the number of design variables, functional requirements, constraints, sub-assemblies, etc.
3. The coupling between elements.

They extended their definition to the design problem, the design process, and the design artifact. We believe that solvability must have to do with cyclic dependencies or cyclomatic complexity because they produce emergence effects that cannot be predicted: by presenting empirical results Watson and McCabe (1996) reported that the number of errors in the implemented (software) systems has been in direct proportion with the cyclomatic complexity. In addition they suggested cyclomatic complexity as a test and evaluation mean.

3.4.2 Complexity Measures for the Design Process

Determining the complexity of a design process may be useful for selecting between two or more design processes for the same design problem. The design problem's complexity does not change, but the complexities of the available design processes may be different. The process complexity measures have applications in design automation where the machine needs to decide between several available processes.

Ko et al (2007) presented an evolutionary complexity of information for analysis of the design process. They reported that the minimization of their entropy based measure of complexity can lead to the identification of the least biased sequence of activities. This thesis does not address the sequencing or

synchronization of the design activities based on their dependencies. Instead we suggest tackling the process complexity at the problem decomposition stage. This is based on the recognition that the design process may be modeled graphically to closely mirror a problem's graphical representations. This trend of modeling started with Simon's definition of decomposability. Simon (1969) defined a complex system as a system of a large number of parts that interrelate in a non-simple manner. Within complex systems, a distinction may be made between the relationships between subsystems and within subsystems. This leads to the concept of decomposability of the complex system. Generally, links within subsystems must be stronger than links between subsystems, allowing for handling the subsystems concurrently rather than as a single system. Simon (1969) proposes "nearly decomposable systems" as an approach to mitigate the effects of complexity in system synthesis.

Chen and Li (2005) presented a complexity index that was the ratio of the complexity of the problem after decomposition to that of the problem before decomposition. They associated this index with the process efficiency. The problem was modelled as the set of design components or physical constituents of a design (design variables), and the design attributes that described the behavioral properties of a design (functional response). They studied the decomposition of the incident matrix of components-attributes pair that reflects how components influence attributes. The immediate observed drawback with this approach is that it does not include the interactions of components-components or attributes-attributes. They presented the following measure for the problem complexity (before decomposition):

$$COM_0 = m \ln(2^n) \tag{8}$$

Where *m* is the number of attributes and *n* is the number of components. The complexity after decomposition had two sources contributing to the total complexity of the interaction-involved matrix: the interaction part and the blocks (the resulting, subsystems, or sub-problems):

$$COM = m_a \ln(2^{na}) + \sum_{i=1}^{nb} m_i \ln(2^{nc_i}) \quad (9)$$

Where m_a is the number of attributes present in the interaction part that is a number between *2* and *m*, *nb* the number of blocks, m_i the number of attributes inside the blocks, and nc_is the number of components inside each block. The first term in Equation (9) is the complexity of the interactions and the second term is the sum of the complexity of the blocks (Figure 3.8).

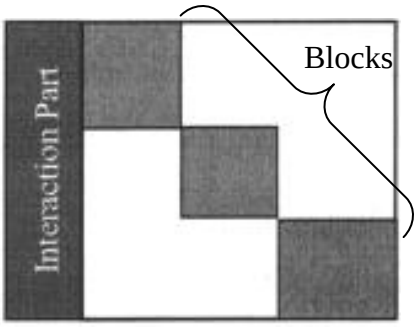


Figure 3.8 Shows the block and the interaction part of a decomposition of an incident matrix. From Chen and Li (2005).

Chen and Li (2005) also presented an iterative algorithm that could determine the number of blocks. They claimed that this approach to mitigate the complexity of the design process has not been reported in the literature before. Our approach to measure the process's complexity is very similar to this, with the distinction that our measure of decomposition has a more holistic approach and does not sum up the complexity of the blocks and interactions to arrive at the complexity of the decomposition. We contend that they incorrectly argue that decomposition can decrease complexity which is perhaps the consequence of their reductionist approach. We will show that regardless of the decomposition type, it cannot reduce the overall complexity of the process, which is utterly rational when the 'no free lunch theorem'[17] is considered.

[17] No free lunch theorem is discussed in the context of optimization theory and states that if a search algorithm achieves superior results on some problems, it must pay with inferiority on other

3.4.3 Complexity Measures for a Design Artifact

Complexity measures for a design artefact are defined for the solution space and thus can be regarded as goodness of design measures. Pahl and Beitz (1985) offered a simplicity rule for embodiment design. This rule is based on the presumption that simple designs are preferred to complex designs. With respect to evaluating the simplicity (or conversely the complexity) of a design artifact, Pahl and Beitz (1985) suggested counting the number of functions represented, evaluating the working principles (number of processes, components, and coupling), or checking the symmetry of shape, topology of shape, motion (easy to analyze and manufacture). No details are provided with respect to analyzing the complexity measures, rather qualitative interpretations by the designer with limited enumeration counts of parts, assemblies, interfaces, etc. are suggested. Balazs and Brown (2002) offer an approach for design artifact simplification through analogical reasoning by reducing redundant sub-graphs. Several of the measures introduced previously including those of Braha and Maimon (1998), and El-Haik and Yang (1999) are also applicable to the design artifact.

3.5 Methodology: Measuring the Complexity of PPO

This thesis assumes that what make a system complex are the three components of size, coupling and cycles. A graph theoretic complexity measure[18] is presented in the Appendix (Section A.1) that is an increasing function of the three components of complexity. In this thesis, this measure is applied to various types of DSM in a unified manner and by that we accomplish the immunization of complex PPO against catastrophic failures. The following discussion presents some necessary definitions and metaphors used in the complexity management and complexity based design method introduced by Marczyk (2008). This method is at the heart

problems. We argue that decomposition makes a problem tractable at the price of more overall complexity.

[18] This measure is the proprietary complexity measure of Ontonix s.r.l and has been commercialized in Ontospace™ software.

of the software, Ontospace™, which its underlying principles have been a knowledge source to the approach of this thesis. Although Ontospace™ was originally devised as a Computer Aided Engineering *analysis* tool, its principles has never been taken as the backbone of a DSS which can facilitate design *synthesis* as well as *analysis*. The conceptual DSS presented in Chapter 8 is a demonstration of this.

Fitness Landscape (FL): FL is a multi dimensional data set of inputs and outputs of a system. Figure 3.9 shows two of the anthills related to a FL. FL is another name for parametric problem space.

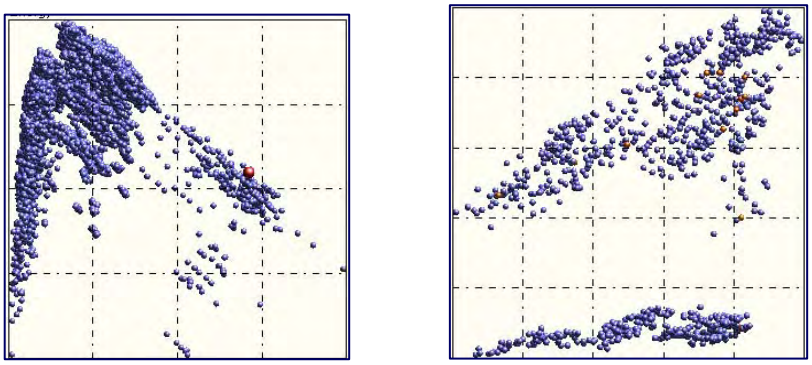


Figure 3.9 Two scatter plots of a FL.

Forming a FL: FL can be formed from experimental data about a system, and/or statistical simulation of the system by means of methods such as Monte Carlo simulation techniques. Combinatorial methods can also be employed. For example a combination of Design of Experiments and Monte Carlo Simulation would be quite adequate for the simulation of product and process models.

Fuzzification of the FL: Ontospace™ fuzzifies the entire FL in 3, 5, or 7 fuzzy levels. Figure 3.10 shows two anthill plots of the fuzzified FL with 5 fuzzy levels.

Fuzzy States: If the FL is partitioned into 3, 5, or 7 fuzzy states, then each multi dimensional partition is a fuzzy state of FL. Fuzzy states are the fuzzified points of the FL.

Map of the System: is the graphical representation of the structure or dependencies in the system (Figure 3.10). Dependencies are determined by the entropy based correlation coefficients.

Complexity of the Map: reflects the coupling, size and cycles of the system. We will refer to the complexity of this map as self complexity.

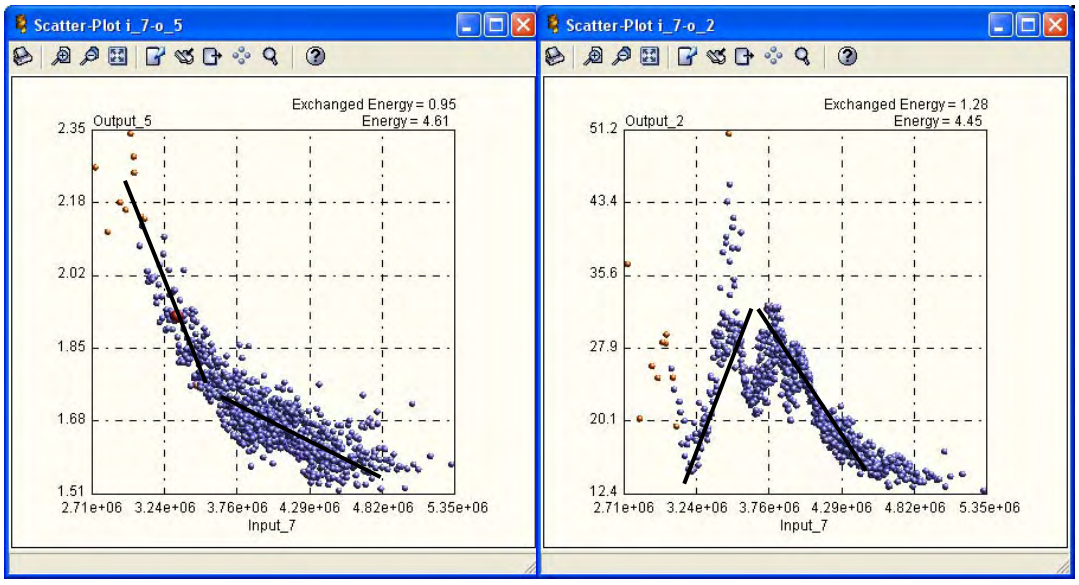


Figure 3.10 Shows different linear trends in the FL.

Lower Complexity Bound: A system with a complexity lower than this bound has lost its intrinsic characteristics and has failed to emerge as a spontaneous self. Dembski (2002) explained that, a system is irreducibly complex when the removal of any of the parts or the links amongst them parts lead to the failure of the system in performing the assumed basic functions. The lower complexity bound represents the irreducible complexity of the system that contains the intrinsic characteristics of the system.

Upper Complexity Bound: The complexity of the system may be increased to this bound without exhibiting chaos. Every closed system can only evolve/grow to a specific maximum value of complexity which is also known as the system's critical maximum complexity. For closed systems, the increase of entropy leads

to the increase in complexity but only to certain point (the upper bound), beyond which even small increase of entropy cause the reduction in complexity and the structure of the system starts to collapse (Marczyk and Deshpande, 2006).

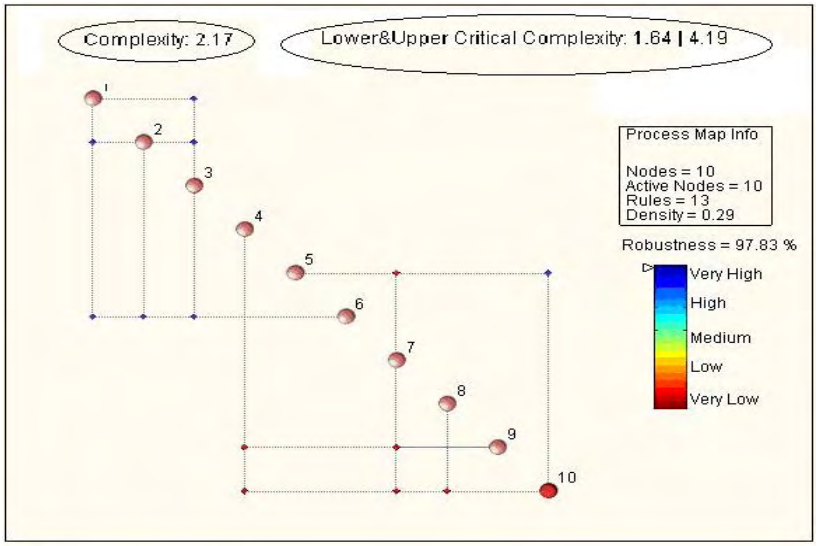


Figure 3.11 An example of a self map and its three complexity measures delivered by Ontospace^{TM}.

Complexity Modes: A mode represents the characteristics of various regions of the FL with certain complexity value complexity modes (or modes) are fixed topologies of a system's important parameters and their correlations. A complexity mode is a collection of system fuzzy states that have a unique map. Figure 3.12 shows a FL and two of its complexity modes with their maps (left and right). The large red and blue points are respectively the input and output parameters that have at least one important correlation to the other variables. The small red dots represent an important link between the two variables. The number of important links and variables that have at least one link to other variables is different for different regions of the FL. Each region with a fixed topology of important variables and links is a complexity mode. A FL may have many modes (e.g. 10). Complexity modes exist because of either nonlinearity or piecewise linearity in the system (Figure 3.9 shows an example of the latter). Each complexity mode is characterized by a different complexity value.

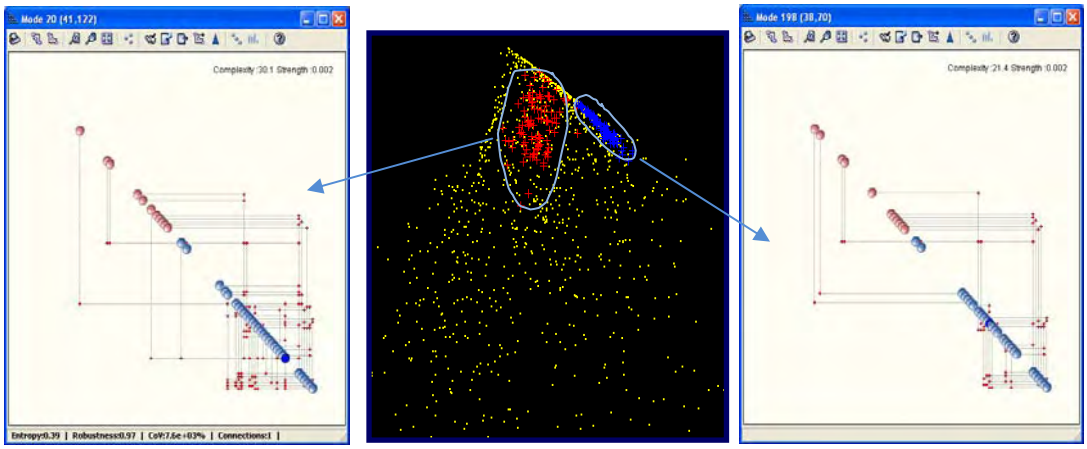


Figure 3.12 Two modes of a FL(left and right).

This methodology (comprising the above definitions) can be best described as a modeless approach for systemic studies. This methodology does not attempt to make models from the observations and then try to estimate the behaviour of the model under different variations in variables (which is the common approach of science). Rather, we use the measurement of complexity as a quantitative indicator of *surprise*. Consequently this measure *per se* can be used in robust decision making with regards to choosing between the solution alternatives.

4 A Template for Complex Design Problems

This Chapter presents our template for engineering complex products. This template is the extended version of Reidsema's model and has characteristics of evolutionary and spiral processes. The template is a simulation based design methodology. Here, the role of simulation is central in immunizing PPO with the help of complexity measures. The template is depicted in Figure 4.1.

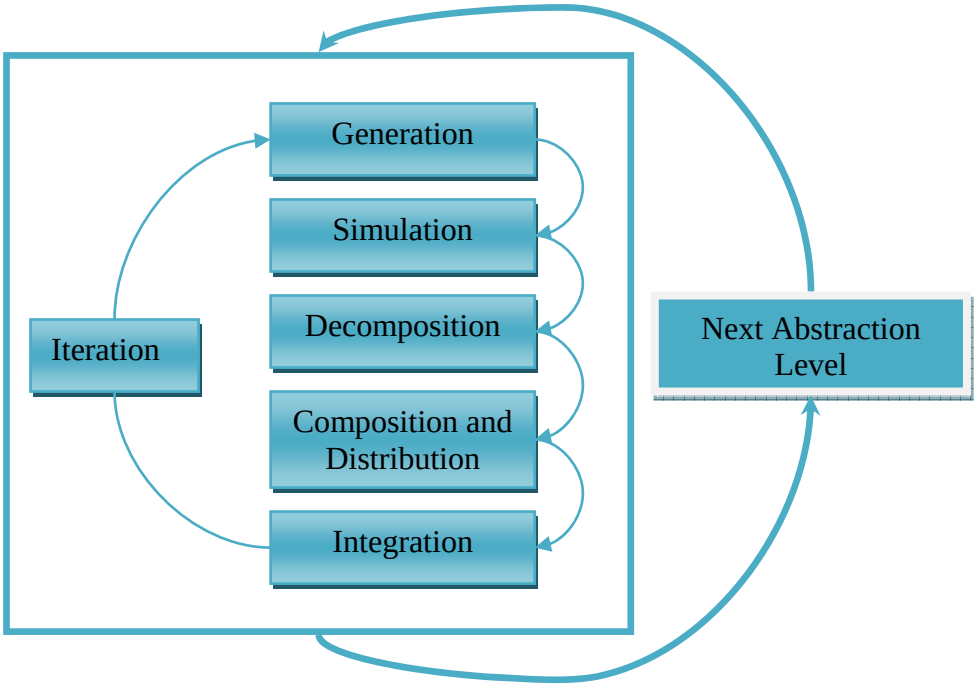


Figure 4.1 A Simulation Based model of design process for complex problems.

It should be noted that the processes at various abstraction levels may be carried out simultaneously. The DSS presented in Chapter 8 exploits and enables this simulation based model of the design problem solving process. The steps of this process are explained next.

4.1 Generation

This is to generate the design concepts, or conceptual design. Here we assume that these concepts are generated in the parametric format as sets of new design variables. But if this is not the case the concepts must be parameterized before proceeding to the next stage. The task of determining the values of the design variables constitute a low level problem solving activity.

Naturally the generated variables at the initial abstraction levels are taken as those variables related to the higher abstraction levels that encompass the more intrinsic characteristics of the product, which has to do with the main functions and performances of the product. In the same way the lower abstraction levels usually locate the variables that describe the detailed functionalities of the product. However these rules may be forsaken. We believe that it is advantageous to think of the variables in the first abstraction level in the hierarchy as a seed that all the solutions of all other abstraction levels, depend heavily upon. Given this premise, the seed must have a foretaste of other problems at other abstraction levels. Thus the seed must not only reflect the most important and general functions of the product but also those functions that are low in the functional decomposition chart and still have large dependencies on the other functions/attributes of the product.

The number of variables at each abstraction level can be regarded as the termination criteria. This is so since more variables implies a larger solution space and a higher number of states in the FL. To follow the ideal approach of Liu et al (2003), we suggest that the number of variables at each stage or abstraction level must increase until about the midpoint in the design process and by then the global trend (determined by the number of variables) must be towards convergence (decrease in the number of variables).

4.2 Simulation

Referring to the process of systematically testing ideas early in NPD as enlightened experimentation, Thomke (2001), in the article "*Enlightened Experimentation: the New Imperative for Innovation*" argued that simulation technologies enhance the number of design breakthroughs by testing a greater variety of ideas in a virtual environment. According to Thomke (2001) "computer simulation doesn't simply replace physical prototypes as a cost-saving measure but it introduces an entirely different way of experimenting that invites innovation." Simulation is the key to resolve performance as well as operational requirements improvement with sensible development and production costs, times and risks for multi-disciplinary systems (Formica and Marczyk, 2007; Sinha et al, 2001). Monte Carlo Simulation is often suggested as means of establishing a design space and FL since creating high-fidelity simulation models are often expensive (Marczyk, 1999; Sinha et al, 2001).

Monte Carlo Simulations can digest information gained from the design of experiments to tune the simulation for higher compatibility with the real system. Monte Carlo Simulation requires the estimation of the conditional probability distribution of every pair of design variables, e.g. from the Design of Experiments results or from the available models of the artefact/problem. Figure 4.2 shows the typical modules that a simulation engine might contain. Here the simulation is the parametric simulation in the statistical sense and the simulation consumes the parametric model of the problem/artefact. Thus a simulation engine must contain a parameterization module to present the problem in the parametric format.

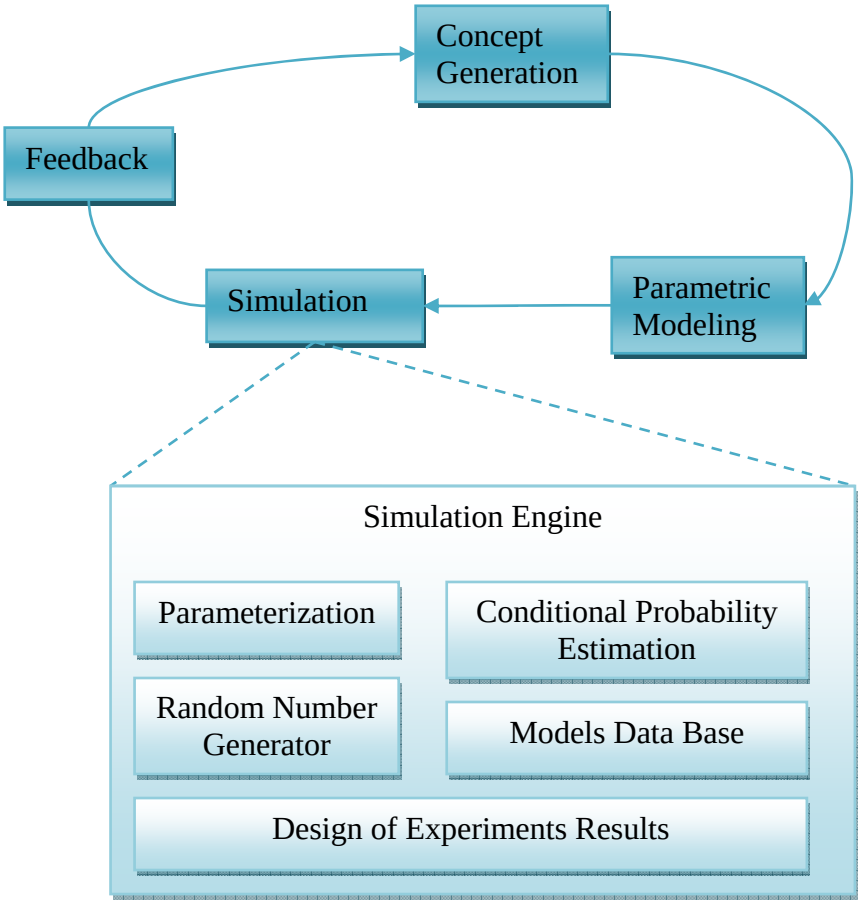


Figure 4.2 Simulation Engine module and its integration in the problem solving procedure.

As stated before the outcome of simulation is the FL or design space at a given abstraction level. A FL is thus the Meta-Model of the design variables and design relations. A Meta-Model is a scatter plot of two random variables (Figure 4.3). Marczyk (1999) described Meta-Models as the goldmines of information; they are ontologies that describe a system (Efatmaneshnik and Reidsema, 2007.a). Any model would not be as precise as the Meta-Models and this forms the basis for the argument that complex systems require Meta-Modelling (Marczyk, 1999). Meta-Models have the ability to transfer all the required information about the structure (and thus the self) of the system and therefore may be used to measure the complexity. Despite the fact that Meta-Models only show the variability of two variables relative to each other (both of which are characteristics of the environment of the system or non-self), the self of the system is embedded or hidden within them in the most prime way.

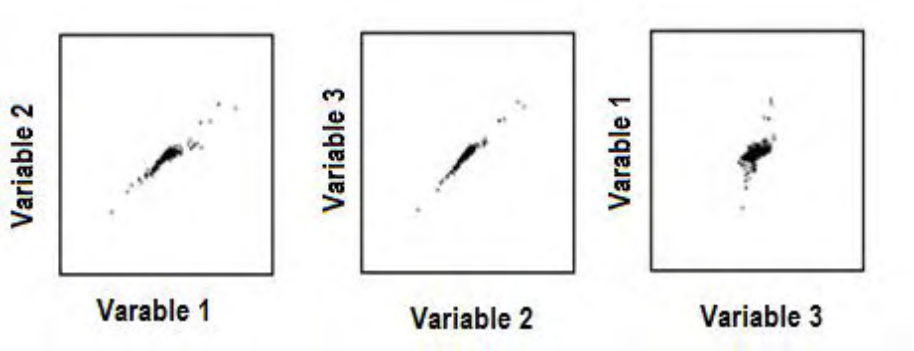


Figure 4.3 Scatter Plots as Meta-Models can be established by Monte Carlo simulation and used for determining the correlation between design variables, from McDonald and Mavris (2000).

We propose the employment of design space simulation in the design process to estimate the PDSM of the product/problem at the upstream of the design process (Efatmaneshnik and Reidsema, 2008.a). The simulated PDSM gives important insights about the couplings in the problem structure and thus the tasks structure. Therefore, by simulation not only the performance of the product can be estimated but also the *design process* can be managed more effectively and efficiently (Efatmaneshnik and Reidsema, 2008.a). In Section 4.4 this issue is extensively elaborated. We suggest that decomposition must be taken as the decomposition of simulated PDSM. Table 4.1 shows an example of a typical simulated PDSM. Figure 4.4 shows the corresponding graph to PDSM in Table 4.1 to which we refer as self graph (or map) of the problem (this graph is identical to the hypergraph presented in the Figure 3.11). Self maps intuitively convey the level of coupling in a system. Decomposition is applied to the self map of the system (Efatmaneshnik and Reidsema, 2008.a).

Table 4.1 A Simulated PDSM is a weighed adjacency matrix. This PDSM has 10 Variables.

-	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
V1	0	0.76	0.45	0.16	0.22	0.77	0.12	0.01	0	0
V2	0.76	0	0.11	0.65	0.44	0.78	0	0	0	0.18
V3	0.45	0.11	0	0.64	0.11	0.31	0.02	0	0.15	0
V4	0.16	0.65	0.64	0	0.45	0.34	0	0	0	0
V5	0.22	0.44	0.11	0.45	0	0	0	0.01	0	0.01
V6	0.77	0.78	0.31	0.34	0	0	0	0	0	0
V7	0.12	0	0.02	0	0	0	0	0.2	0.7	0.1
V8	0.01	0	0	0	0.01	0	0.2	0	0.2	0.8
V9	0	0	0.15	0	0	0	0.7	0.2	0	0.9
V10	0	0.18	0	0	0.01	0	0.1	0.8	0.9	0

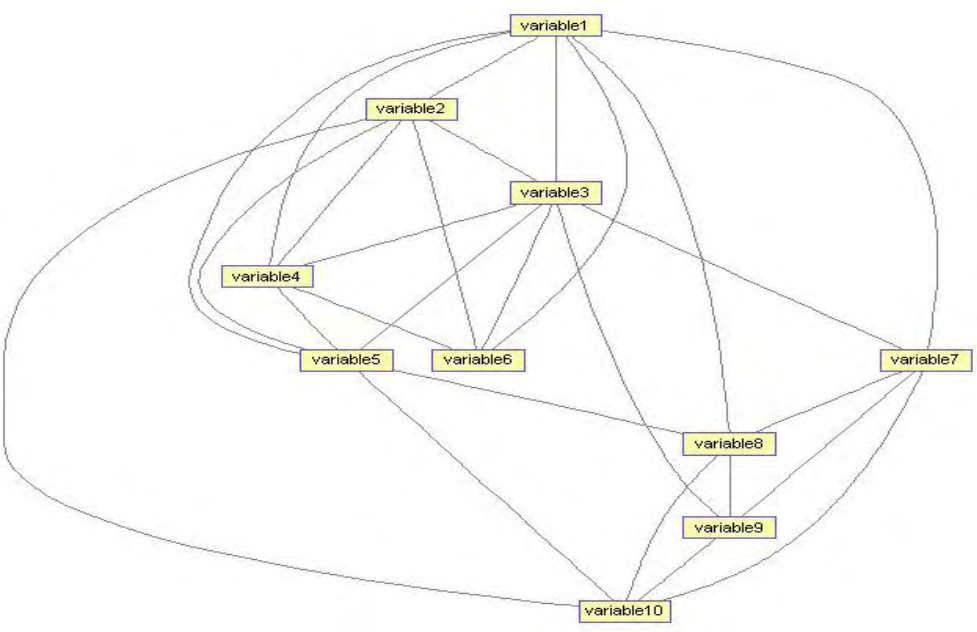


Figure 4.4 The self graph of problem/system.

4.3 Decomposition

According to Papalambros (2002) decomposition of large-scale design problems allows for:

- conceptual simplification of the system
- reduction in the dimensionality of the problem
- more efficient computational procedures
- utilization of different solution techniques for individual sub-problems
- simultaneous design, modularity, multi-objective analysis
- efficient communication and coordination among the diverse groups involved in the design process

Problem decomposition and partitioning of the self map of the system fits within the area of graph partitioning. A bi-partitioning of graph *G* is a division of its vertices into two sets or sub-graphs, P_1 and P_2. Similarly a *k*-partitioning is the division of the vertices of the graph into k non-empty sets $P = \{P_1, P_2, \ldots, P_k\}$. A graph can be partitioned in many different ways. In the domain of problem solving, every node or vertex of a graph represents a variable of the system and every edge of the graph suggests that two parameters of the system are dependent on each other; however there are several other representational schemes (for example see Michelena and Papalambros (1995), and Chen et al (2005)). Here, the strength of the relationship between two variables is the corresponding edge weight. An undirected graph as the self map of the system indicates that variables affect one another mutually and equally. The sub-graphs can be regarded as subsystems or sub-problems or agents (Kusiak, 1999). The notion of agency implies that the sub-problems are solved more or less independently from each other. Each design team has autonomy to explore parts of the solution space that is of interest to its own assigned sub-problem (agent).

We suggest decomposing the PDSM in several modes and according to its connectivity level (Efatmaneshnik and Reidsema, 2008.a). Problem connectivity

is the total number of edges in the self map of the product/problem divided by the total number of possible edges; that is the number of edges of a complete graph with the same number of nodes (Efatmaneshnik and Reidsema, 2008.a). The total number of possible edges in a complete undirected graph with n nodes or vertices is:

$$K = \binom{n}{2} = \frac{n \times (n-1)}{2} \tag{2}$$

If the self map of the PDSM has *k* connections (edges), we define the problem connectivity as:

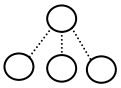
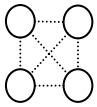
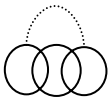
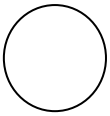
$$p = \frac{k}{K} \tag{3}$$

The decomposition modes have been scattered in the literature (Sosa et al, 2000; Klein et al, 2003.b; Browning, 2001) and are brought together here for the first time (Table 4.2). Bearing in mind that it is usually desirable to have subsystems of similar order, the implementation of some of these decomposition modes (in particularly full decomposition mode and integrative mode) may not always be feasible. The connectivity values in Table 4.2 are based on this knowledge and the experience of the author with randomly generated graphs. For problems with a denser self map (higher connectivity), modular clustering and overlap decomposition can be used. If the problem's map is very dense and the system is regarded as highly complex then it may not be decomposed at all (Bar-Yam, 2004). We will come back to this important issue in Chapter 5.5.

Each of these decomposition modes brings specific strengths, weaknesses and particularity to the problem solving process. As an example, an aircraft with a blended wing body may not be decomposed completely into separate body and wings with the related design variables being independent or loosely dependent (Figure 4.5). Instead for systems that have subsystems with fuzzy boundaries overlap decomposition may be used. The reason that much effort in this thesis has been devoted to problem decomposition is that clustering in fact is the key to tying top-down, activity-based DSMs together with bottom-up, parameter-based

DSMs (Browning, 1999). The decomposition problem will be treated in Chapter 5.

Table 4.2 Decomposition Modes of self map of problems.

Connectivity	*Very Low (0-0.02)*	*Low (0.02-0.1)*	*Intermediate (0.1-0.2)*	*High (0.2-0.3)*	*Very High (0.3-1)*
Possible or best decomposition strategy	Full decomposition	Integrative Clustering	Modular clustering	Overlap clustering	No decomposition
Illustration					

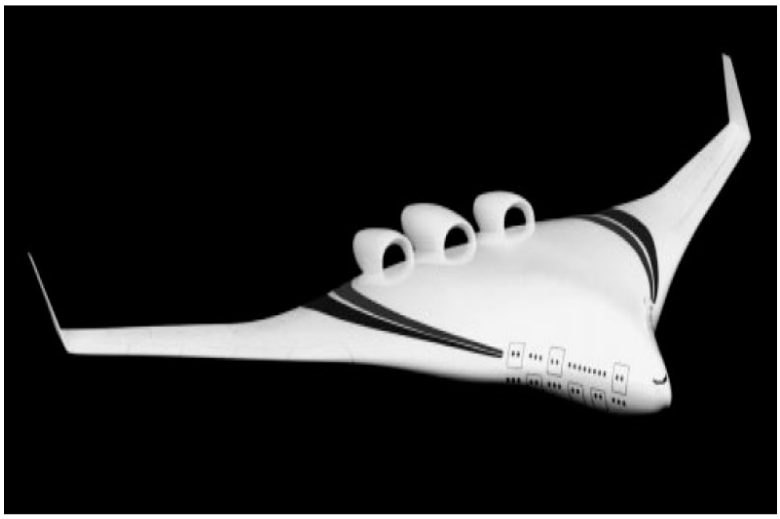


Figure 4.5 Two components (subsystems) are overlapped in blended wing-body types of aircrafts.

4.4 Distribution (and Composition)

Distribution refers to the distribution of the design tasks amongst the design teams and composition is thus the formation of the design teams. Integrated NPD describes how tasks are interconnected and seeks to integrate the product process and organization (Prasad, 1996). As depicted in Figure 4.6, organizational

partitioning and integration are tied to the nature of the product decomposition (Gulati and Eppinger, 1996; Browning, 1999; Eppinger and Salminen, 2001). It is arguably true that the efficiency of the design integration process is somehow dependent on the alignment of the task structure and organizational structure (Browning, 1999).

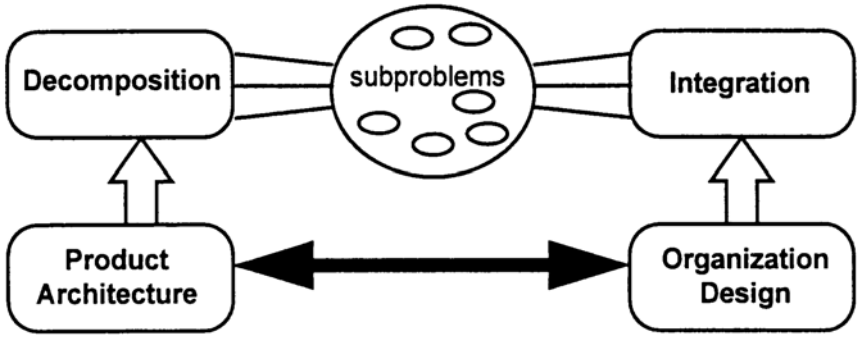


Figure 4.6 Product architecture is tied to organization through decomposition/integration problem, after Gulati and Eppinger (1996).

Thus integrated NPD requires that decomposition and integration schemes have congruence. This harmony marks a complex NPD project with success. In general two main approaches are reported in the literature for the deliberate alignment of problem structure and NPD organization structure (Efatmaneshnik and Reidsema, 2008.a):

1. *Bottom up planning for flexible organizational structure* forms the design teams after product decomposition has taken place. For example the aligned organizational architecture in charge of conceptual and parametric design of an aircraft's body and wings, in the same way, constitute two different design teams with overlapping boundaries. Tanaka et al (2000) took this approach in the context of distributed problem solving and called it multi-agent system creation. This approach has also been used in a manufacturing system MetaMorph (Maturana et al, 1999) that could dynamically change its form to mimic the task structure. In this case the

number of the subsystems the system is decomposed to may be maximized for better overall performance.

2. *Top-down planning for fixed organizational structures* decomposes the problem/product in a way that suits the organizational structure; this is used when the organizational structure is fixed and solid which means that design teams are formed prior to the introduction of the problem. A greater use of coordination activities between the design teams and/or the use of integration teams results. In this case the number of the subsystems is determined according to the number of design teams.

These two approaches correspond to using low level and high level knowledge for design planning. In general in a problem solving environment the designers actions can be planned or controlled by using three kinds of knowledge (Reidsema, 2001) (Figure 4.7):

1. Low level problem knowledge
2. Medium level knowledge of the problem solving process
3. High level organisational knowledge

Browning (2001) emphasized that parameter-based DSM which represents the low level product knowledge has integrative applications:

> Most design planning takes place in a top–down fashion through decomposition. If they begin at the top, such models rarely reach the lowest levels of design activity, where individual design parameters are determined based on other parameters. A bottom-up, integrative analysis of these low-level design activities can provide process structure insights.

This characteristic of the parameter based DSM which represents the low level product knowledge makes it suitable to be utilized in planning and distributing complex engineering problems (Efatmaneshnik and Reidsema, 2008.a).

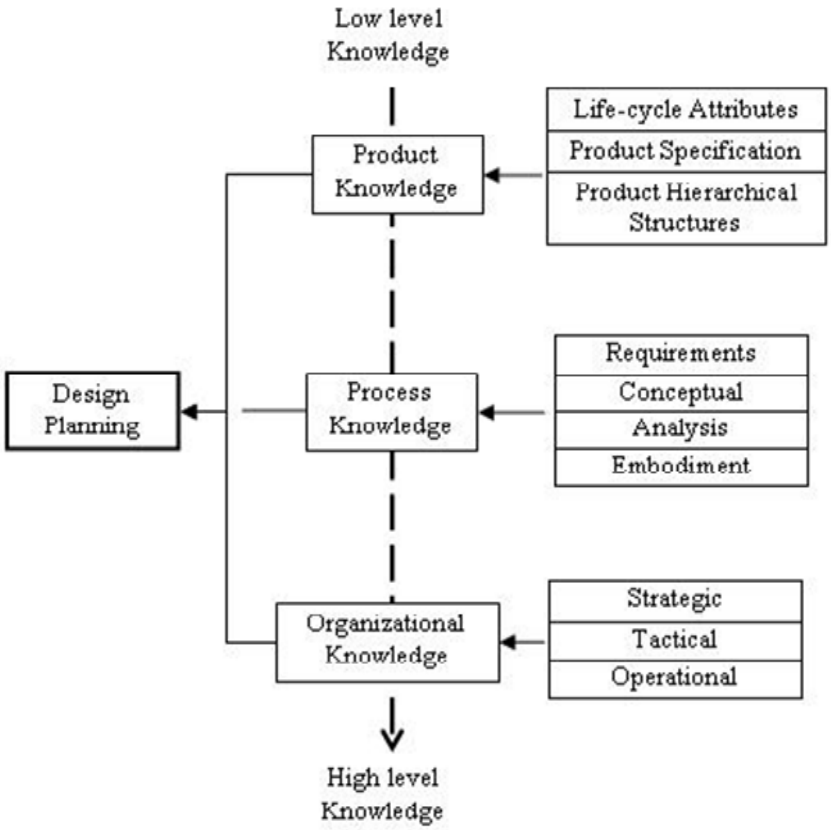


Figure 4.7 Design Process Knowledge (Reidsema, 2001).

4.5 Integration

Integration combines the partial solutions of a large problem (Reidsema, 2001). The integration problem of complex problems and complex problem solving environments is the main challenge for problem solvers. For complex systems, due to coupling between the distributed tasks, integration may not be performed linearly simply by adding the partial solutions together. Since the coupled problems tend to be nonlinear (same as the coupled differential equations) the solutions may not be achieved by using the usual concurrent planning (that adds the partial solutions to obtain the overall solution). The nonlinearity limits the kind of knowledge being used for planning.

In product design the performance and operational requirements of the product are micro or low level parameters, whereas production costs, times and risks are macro or high level and often emergent properties of the process

(Efatmaneshnik and Reidsema, 2009). Macro level properties (those that are observed at the organizational level) dynamically arise from the interactions between the micro-level properties that are the low level activities (Efatmaneshnik and Reidsema, 2009). Thus, in order to resolve the uncertainty and dealing with high level emergent properties of the process and organisational operations (that can be chaotic), the low level knowledge of the problem must be used to characterize the behavioural rules of individual problem solvers (Efatmaneshnik and Reidsema, 2009). Using low level knowledge of the product is indeed a bottom up approach that has been suggested as the panacea for complex problem solving failures (Anderson, 2006), (Efatmaneshnik and Reidsema, 2009). Bottom up modelling of the design process offers the greatest fidelity to the process being modelled or emulated (Anderson, 2006).

Integration within the design process can be conducted by two major methods (Efatmaneshnik and Reidsema, 2008.a):

1. Supervised integration
2. Unsupervised integration

Supervised problem solving architecture involves high level integration teams and centralized planning (Efatmaneshnik and Reidsema, 2008.a) (see Figure 4.8). Eppinger (1997) stated that:

> One important level of integration takes place within each development team; this is the now common practice of concurrent engineering, in which a cross-functional team addresses the many design and production concerns simultaneously. To assure that the entire system works together, sub-system development teams must work together and for that additional teams are assigned the special challenge of integrating those subsystems into the overall system.

However for densely coupled and complex problems/systems using high level integration teams as coordinators cannot be effective, since the high load of coordination complexity would be a greater barrier to effectiveness of the integration process.

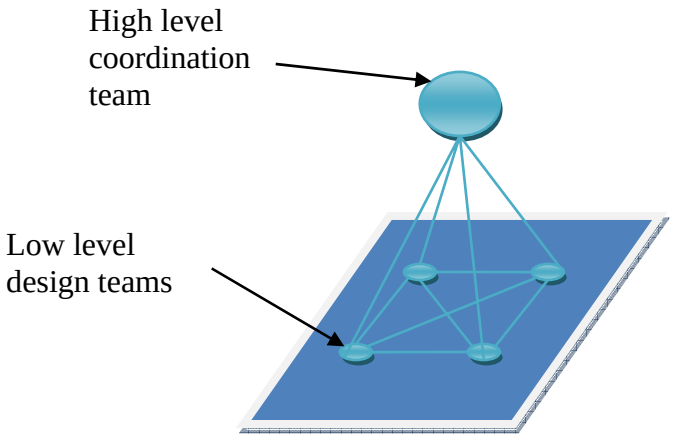
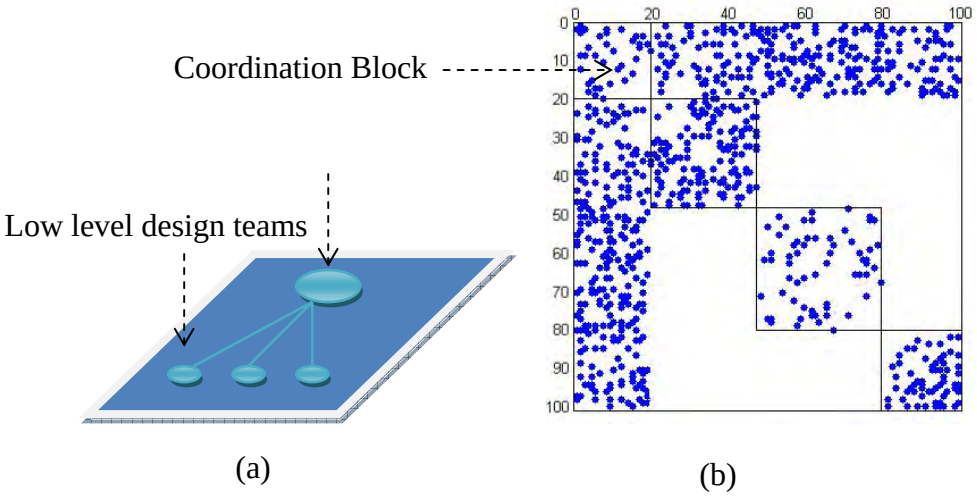


Figure 4.8 Integration team acts as a high level coordinator.

Unsupervised problem solving architectures can cope with problems of more complexity using a distributed planning approach (Efatmaneshnik and Reidsema, 2008.a). These include systems that use 1) low level integration team and 2) multi agent architecture and 3) information intensive architecture (Efatmaneshnik and Reidsema, 2008.a).

Systems using low level integrators and multi agent architectures correspond to two decomposition patterns that are recognized by Sosa et al (2000) as coordination-based and modular. Coordination based decompositions partition the system into several relatively independent subsystems and only one (or few) strongly connected subsystem(s) namely the coordination block(s) (Figure 4.9). The identification of coordination block (Figure 4.9(b)) in a system can be performed through integer programming (Sosa et al, 2000). The coordination block (C_C) is an integrative subsystem and the design team in charge of integrative subsystem design is regarded as a low level integration team that implicitly coordinates the activities of other teams (Efatmaneshnik and Reidsema, 2008.a). Obviously, the design of the integrative subsystem must be much more

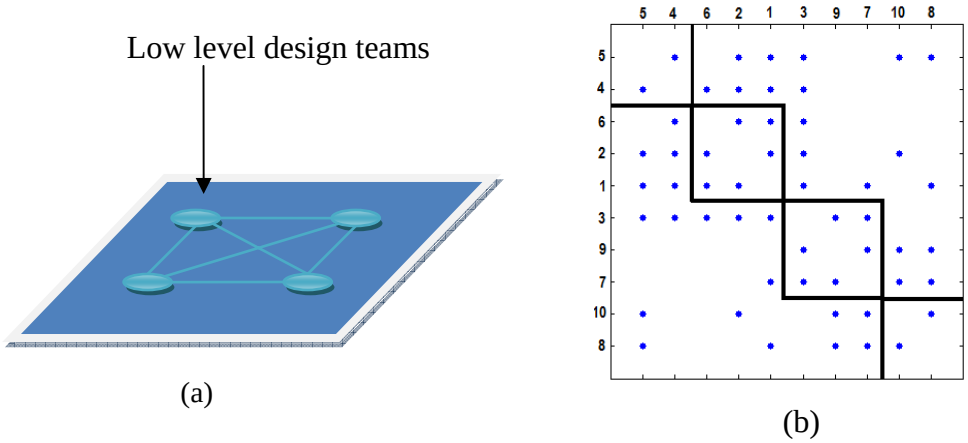
complex than the other subsystems. As such, the integration of the complex systems with more than a certain amount of coupling is not desirable with low level integration schemes through coordination based problem decompositions (Efatmaneshnik and Reidsema, 2008.a).



(a) (b)

Figure 4.9 Interactions between design teams of low level integration scheme (a) and the corresponding PDSM of order 100 with coordination based 4-partitioning (b).

The interactions between the design teams in multi agent systems are autonomous and based on agents social knowledge (Efatmaneshnik and Reidsema, 2008.a). Multi agent systems are a relatively complex field of research. The solution to the design problems in multi agent systems is formed in a self organizing fashion that emerges as result of autonomous interaction of the agents (Figure 4.10(a)); multi agent systems correspond to modular problem decomposition (Figure 4.10(b)) (Efatmaneshnik and Reidsema, 2008.a).

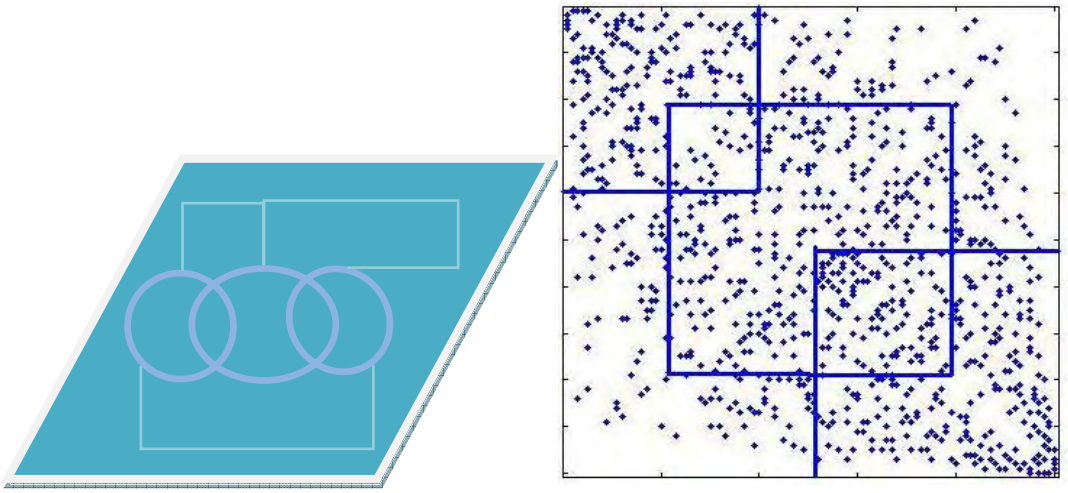


(a) (b)

Figure 4.10 Multi agent design system (a), the corresponding modular PDSM decomposition (b).

Information intensive architecture can also be regarded as multi agent system in which the design teams (or coalition of agents) have overlapping boundaries (Efatmaneshnik and Reidsema, 2008.a). Information intensive architecture corresponds to overlap decomposition of product/system in which subsystems are overlapped and share some of the design variables with each other (Efatmaneshnik and Reidsema, 2008.a). Information intensive structures facilitate collaborative design for large scale and complex design problems.

According to Klein et al (2003.b) collaborative design is performed by multiple participants representing individuals, teams or even entire organizations each potentially capable of proposing values for design parameters and/or evaluating these choices from their own particular perspective. For large scale and severely coupled problems collaborative problem solving is possible when the design space or problem space is decomposed in an overlapped manner: the design teams explicitly share some of their parameters, problems, and tasks (figure 4.11). The main characteristic of this process model is the intense collaboration between coalitions of agents making this mode an information and knowledge intensive process (Klein et al, 2003.b). The impact of new information on the design process in this integration scheme is relatively high; as such overlap decomposition and its corresponding integration scheme are suitable for problems of high complexity and self connectivity (Efatmaneshnik and Reidsema, 2008.a).



(a) (b)

Figure 4.11 Design teams (a) as well as product partitions (b) have overlapped boundaries.

Following the idea of Enlightened Engineering (Bar-Yam, 2004) we propose that for problems with high levels of connectivity several design groups must work in parallel to each other on the same problem (that has not been decomposed) (Efatmaneshnik and Reidsema, 2008.a). The design teams in this situation compete rather than cooperate with each other. The innovative problem solving as the integration breakthrough (Bar-Yam, 2004) is discussed and treated in Chapter 6.

4.6 Discussion: Adaptive Structuration

The proposed model of design goes beyond the traditional models of design and is virtually an amalgamation of organization theory and design theory. An important research question in the field of organization design is how to constitute cross-functional teams (Browning, 1999). Coordination schemes are needed to direct the design process so that a design solution is sought in a way that accommodates the team interactions (Chen and Li, 2001). “The productivity of design teams depends to a large extent on the ability of its members to tap into an appropriate network of information and knowledge flows” (Kratzer, 2004).

Utilizing cross-functional teams that adapt the organization structure to the task structure is one way to address these situations (Browning, 1999). The presented template poses that at each round of the design cycle (of each abstraction level) the organization (team configuration) must be adapted to the generated tasks and reconfigured according to the complexity of the problem at that level. According to their adequacy to cope with problems of higher complexity, these integration schemes are listed in Figure 4.12 (Efatmaneshnik and Reidsema, 2008.a). The idea of embedding different knowledge sharing patterns amongst the design agents and design teams of one system has been to date, considered by several other systems researchers (Zhang, 1992; Shen and Norrie, 1998; Rosenman and Wang, 1999; Chen and Li, 2001). These models will be reviewed in Chapter 8.

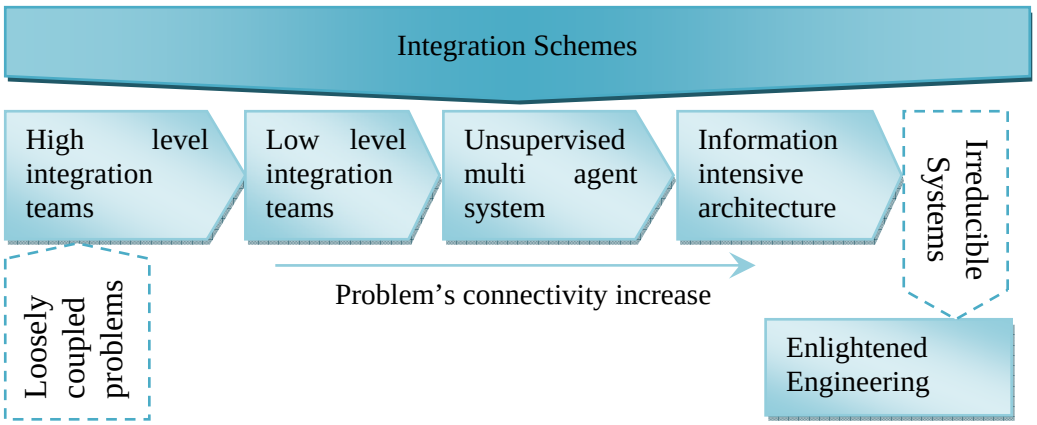


Figure 4.12 Ranking various integration schemes capability in coping with complexity.

"The production and reproduction of the social systems through members' use of rules and resources in interaction" has been studied by Giddens in a sociological context and is known as the *Theory of Structuration* (DeSanctis and Poole, 1994). DeSanctis and Poole (1994) adopted Giddens theory to study the interaction of groups and organizations with information technology denoted as *Adaptive Structuration Theory*. The theory deals with the evolution and development of groups and organizations with observable patterns of relationships and communicative interaction among the people (DeSanctis and Poole, 1994).

IMMUNE, the conceptual DSS presented in Chapter 8 enables Adaptive Structuration through virtual teams. “Virtual teams (or coalitions) are groups of individuals collaborating in the execution of a specific project while geographically and often temporally distributed, possibly anywhere within (and beyond) their parent organization” (Leenders et al, 2003). “Virtual teams work across boundaries of time and space by utilizing modern computer-driven technologies”; as an instrument of team design, Information Technology (IT) is used for creating interdependent relationships by actively shaping and reshaping interdependencies and the communication structure of the virtual teams (Leenders et al, 2003). As these are altered, consequently, so are the team’s productivity (Leenders et al, 2003) and creativity (DeSanctis and Monge, 1999) which are related to the creation of the appropriate information flow between the design teams. Adaptive Structuration can be implemented with less effort by using virtual teams. This is so, because the creation of interdependencies within and between the virtual teams is arguably easier than in a conventional team (DeSanctis and Monge, 1999), and so the management of creativity is more affordable in virtual organizations (Leenders et al, 2003).

5 Immune Decomposition and Process Immunity

Decomposition is a means of reducing the complexity of the main problem to several sub-problems. This is a reductionist approach which reduces complex things to their constituent parts and their interactions in order to understand their nature. In contrast holism is an approach to problem solving that emphasizes the study of complex systems as wholes. “In the design community, decomposition and partitioning of design problems has been attended for the purpose of improving coordination and information transfer across multiple disciplines and for streamlining the design process by adequate arrangement of the multiple design activities and tasks” (Michelena and Papalambros, 1997).

Pimmler and Eppinger (1994) explained that “for a complex product, such as an automobile, a computer, or an airplane, there are thousands of possible decompositions which may be considered; each of these alternative decompositions defines a different set of integration challenges at the organizational level”. Alexander (1964) posed that design decomposition (or partitioning) must be performed in a way that the resulting sub-problems are minimally coupled. In the literature this is also referred to as optimal decomposition (Michelena and Papalambros, 1997) and robust decomposition (Browning, 1999). Along the same line Simon (1969) suggested that complex design problems could be better explained when considered as “hierarchical structures consisting of nearly decomposable systems organized such that the strongest interactions occur within groups and only weaker interactions occur among groups”.

More coupled sub-problems usually lead to more process iterations and rework, because conflicts may arise when dependency (edges) in between the subsystems exists (Efatmaneshnik and Reidsema, 2008.a). A conflict is when the solution to one sub-problem is in contrast with the solutions to another sub-problem(s). An important conflict resolution technique is negotiation. Negotiation leads to iteration in the design process. Obviously a design process with least number of iterations is more desirable, and to do this decomposition must be

performed in way that the entire system after decomposition entails less coupling (see Figure 5.1).

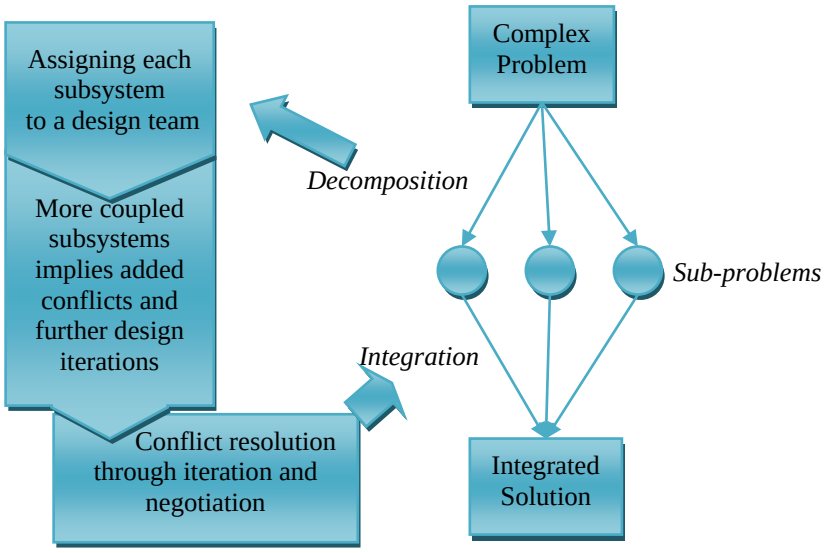


Figure 5.1 More coupled sub-problems increase the number of process iterations.

The system (or problem) is fully decomposable if there is no edge in between the sub-systems. In this case the corresponding design process can be made fully concurrent: problems are solved separately and solutions are added together. A two stage algorithm is usually used to decompose a design problem into sub-problems that are less coupled (Kusiak, 1999; Chen et al, 2005). These stages are:

1. To diagonalize the PDSM of the problem (or the adjacency matrix of the corresponding graph).
2. To cut the diagonalized PDSM from the appropriate points.

5.1 Spectral Diagonalization Technique

Several methods exist for diagonalization including integer programing (Kusiak, 1999), genetic algorithms (Altus et al, 1996) and spectral methods. Spectral graph theory uses the eigenvalues and eigenvectors of the adjacency and

Laplacian matrices. The eigenvectors of adjacency matrix and Laplacians can be used to diagonalize the adjacency matrices of both weighted and un-weighted graphs. Consider A to be the adjacency matrix of an undirected, weighed graph (*G*). An automorphism of a graph *G* is a permutation g of the vertex set of *G* with the property that, for any vertices *u* and *v*, we have ug ~ vg if and only if *u* ~ *v*. "*vg"* is the image of the vertex *v* under the permutation *g* and (~) denotes equivalence. Automorphisms of graph *G* produce isomorphic graphs (Cameron, 2004).

The first step in spectral partitioning of graphs is to sort the eigenvectors of the adjacency and Laplacian matrices in ascending order, and then to permute *G* by those indices of the sorted vector (Efatmaneshnik and Reidsema, 2007.b). Some of these permutations (by different sorted eigenvectors) are diagonalized (Efatmaneshnik and Reidsema, 2007.b). Although initially it was thought that only the eigenvector of the second eigenvalue of the Laplacian (known corresponding as the Fiedler vector and Fieldler value) has the property of diagonalization but later it was shown that using other eigenvectors (of both adjacency and Laplacian) can outperform the Fiedler vector in this regard, specifically in case of the weighted graphs (Alpert et al, 1999). Figure 5.2(a) shows the graphical representation of the adjacency matrix of an un-weighted randomly generated graph of order one hundred. Each dot points to the existence of a link between two corresponding variables. The automorphisms of this adjacency matrix are also shown which are permuted by fiddler vector (Figure 5.2(b)), third eigenvector of the Laplacian (Figure 5.2(c)), and 98^{th} eigenvector of adjacency matrix (Figure 5.2(d)).

The utilization of spectral diagonalization has not been reported in the engineering design literature. One reason for this may be that mathematical representation of this method used in the discrete mathematics community is very different from the way it is presented here. Traditionally the diagonalization was achieved through lengthy integer programming and hefty branch and bound algorithms (Kusiak, 1999). The spectral diagonalization technique is already exploited extensively in the context of discrete mathematics (Alpert et al, 1999; Spielman and Teng, 2007), circuit design (Chan et al, 1994), data mining (Ding et

al, 2001; White and Smyth, 2005) and image segmentation (Shi and Malik, 2000). This algorithm is very fast compared to the traditional integer programming and branch and bound algorithms that were iteration based as those reported in Kusiak (1999).

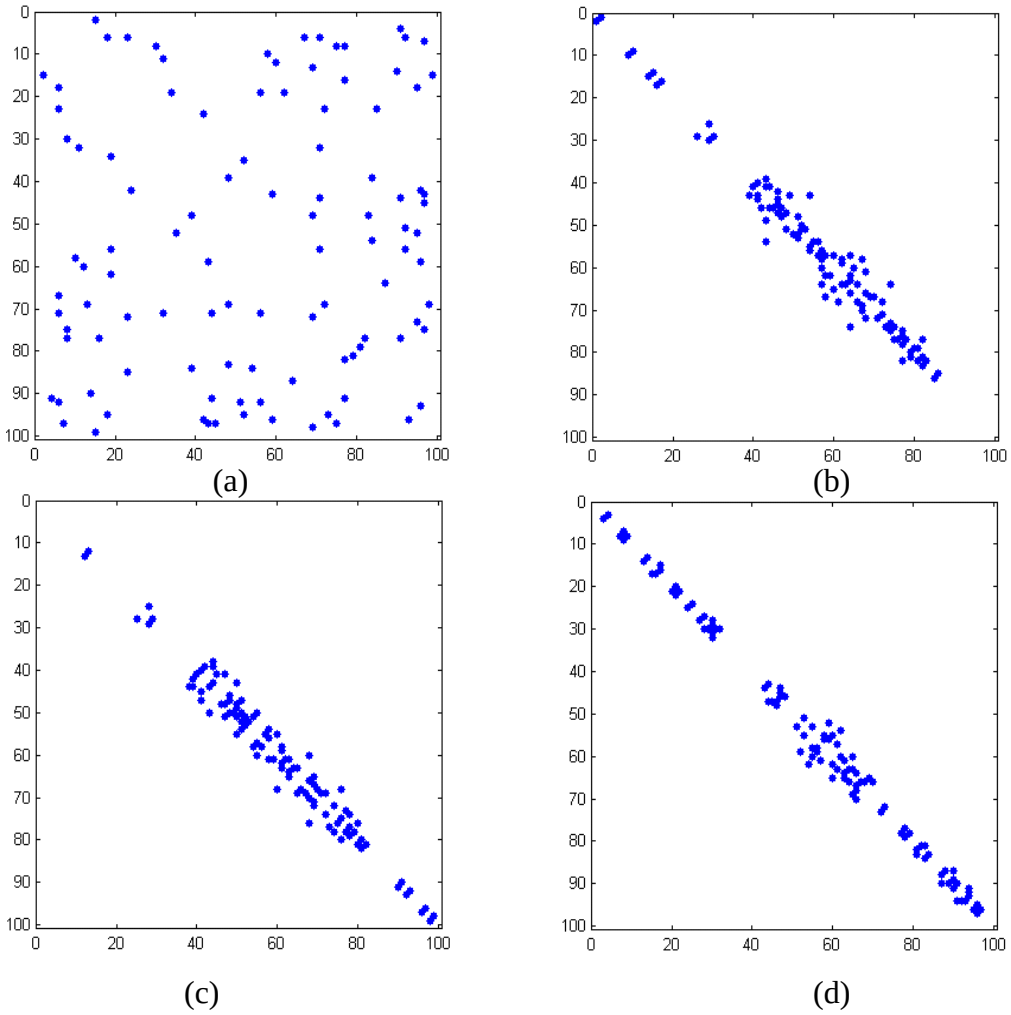


Figure 5.2 Shows the adjacency matrices of a graph (a) and its spectral permutations by various eigevectors that are diagonal automorphisms of the graph.

5.2 Partitioning Quality Criteria

After diagonalization, the cutting points must be determined. Since for a given graph many different decompositions are possible there must be a metric that enables comparison between them (Efatmaneshnik and Reidsema, 2008.a). These metrics are referred to as partitioning quality criteria. Table 5.1 shows some of these metrics and characteristics. In this table *k* denotes the number of sub-graphs and *n* is the cardinality (order) of the original graph, and λ_i is the i_{th} eignevalue of the Laplacian matrix, E_h is sum of the weights of all edges that have only one end in sub-graph P_h:

$$E_h = \sum_{i \in P_h, j \notin P_h} a_{i,j} \quad (1)$$

Also the cut size is defined as:

$$cut(P_r, P_s) = \sum_{i \in P_r, j \in P_s} a_{i,j} \quad (2)$$

And finally the total edge weights in the sub-graph P_h is:

$$E(P_h) = \sum_{i \in P_h, j \in P_h} a_{i,j} \quad (3)$$

For a more detailed comparison in between the performance of these metrics see Chan et al (1994) and Verma and Meila (2003). The minimization of the quality partitioning criteria is an optimization problem and requires employing the appropriate optimization techniques. There are various spectral methods to determine the indices of cut points that can minimize different partitioning criteria however their accuracy is disputed (Verma and Meila, 2003; Alpert et al, 1999). We suggest using exhaustive search algorithm after diagonalization.

Table 5.1 Several partitioning quality criteria.

Name	*The measure*	*Proposed by*	*General remarks*
Cut ratio	$\frac{cut(P_1,P_2)}{\min(\lvert P_1\rvert,\lvert P_2\rvert)}$	Spielman and Teng (2007)	Has a lower and an upper bound: $\lambda_2/2$, $\sqrt{\lambda_2(2d-\lambda_2)}$
Cut Ratio	$\frac{cut(P_1,P_2)}{\lvert P_1\rvert \times \lvert P_2\rvert}$	Cheng and Hu (1989)	Has a lower bound: $\frac{\lambda_2}{n}$
Min-Max cut ratio	$\frac{cut(P_1,P_2)}{E(P_1)}+\frac{cut(P_1,P_2)}{E(P_2)}$	Ding et al (2001)	Favors balance sized sub-graphs
Normalized cut ratio	$\frac{cut(P_1,P_2)}{E_1+E(P_1)}+\frac{cut(P_1,P_2)}{E_2+E(P_2)}$	Shi and Malik (1991)	Favors balance sized sub-graphs
Min cut	$\sum_{h=1}^{k} E_h$	Alpert et al(1999)	Can lead to unbalance sized sub-graphs
Cost	$\frac{\frac{1}{2}\sum_{i=1}^{k} E_i}{\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\lvert P_i\rvert\times\lvert P_j\rvert}$	Yeh et al (2005)	Leads to balance ordered sub-graphs
Scaled cost	$\frac{1}{n(k-1)}\sum_{i=1}^{k}\frac{E_i}{\lvert P_i\rvert}$	Chan et al (1994)	Has a lower bound: $\frac{\sum_{i=1}^{k}\lambda_i}{n(k-1)}$
Modality function	$\sum_{i=1}^{k}\left[\frac{E(P_i)}{E(G)}-\left(\frac{E_i+E(P_i)}{E(G)}\right)^2\right]$	White and Smyth (2003)	Strong cluster identification metric for very large networks. Maximizes at k =3.

To reduce the computational costs it is desirable to have an estimate of the number and order of subsystems although this is not essential. As a general rule, a higher number of sub-problems is better (Michelena and Papalambros, 1995) to the extent that decomposing the problem into its very elements (the single variables) might seem appropriate. This means that the number of sub-problem should is equal to the number of variables. SINE (Brown et al, 1995) for example was a multi agent design system based on the mechanism of assigning single design variables to each of its design agents that were called single function agents. In Section 5.3 a partitioning quality criterion of decompositions is presented that amongst other advantages explicitly suggests the number of partitions that should not be used.

5.3 Real Complexity

Let S be the graphical representation of a problem with the adjacency matrix $A = [a_{i,j}]$ and its complexity $C(S)$ (self complexity) measured by the graph theoretic complexity measure presented in Section A.1. Consider k-partitioning P of the graph S: $P = \{P_1,P_2,\ldots,P_k\}$. Each of these sub-graphs is a block of the system. Let $C(P_i)$ be the complexity of each sub-graph determined by the complexity measure. A block diagram is the graph representation of partitioned graph (Diestel, 2005) i.e. it is more abstract than the self of the system. As such a block diagram is the graphical representation of the decomposed system (Figure 5.3) (Efatmaneshnik and Reidsema, 2008.a). We define the k dimensional square matrix B as the Complexity Based Adjacency Matrix of the Block Diagram with the diagonal entries as the complexity of the sub-graphs (or blocks), and the off-diagonal entries as the sum of the weight of the edges that have one end in each of the two corresponding sub-graphs (Efatmaneshnik and Reidsema, 2008.a). The real complexity of the block diagram $C(B)$ is achieved by applying the complexity measure to matrix B (Efatmaneshnik and Reidsema, 2008.a). This measure is a better measure of complexity of systems after decomposition and a more holistic one than the complexity index of Chen and Li (2005) that was the sum of some components.

$$B = \begin{bmatrix} C_1 & L_{1,2} & \cdot & \cdots & \cdot & L_{1,k} \\ L_{2,1} & C_2 & \cdot & \cdots & \cdot & L_{2,k} \\ \cdot & \cdot & \cdot & & & \cdot \\ \vdots & \vdots & & \ddots & & \vdots \\ \cdot & \cdot & & & \cdot & \cdot \\ L_{k,1} & L_{k,2} & \cdot & \cdots & \cdot & C_k \end{bmatrix} \quad (4)$$

Where $\forall i,j = \{1,\ldots,k\},\ C_i = C(P_i)$ and $L_{ij} = \sum_{r \in P_i,\ s \in P_j} a_{r,s}$ (5)

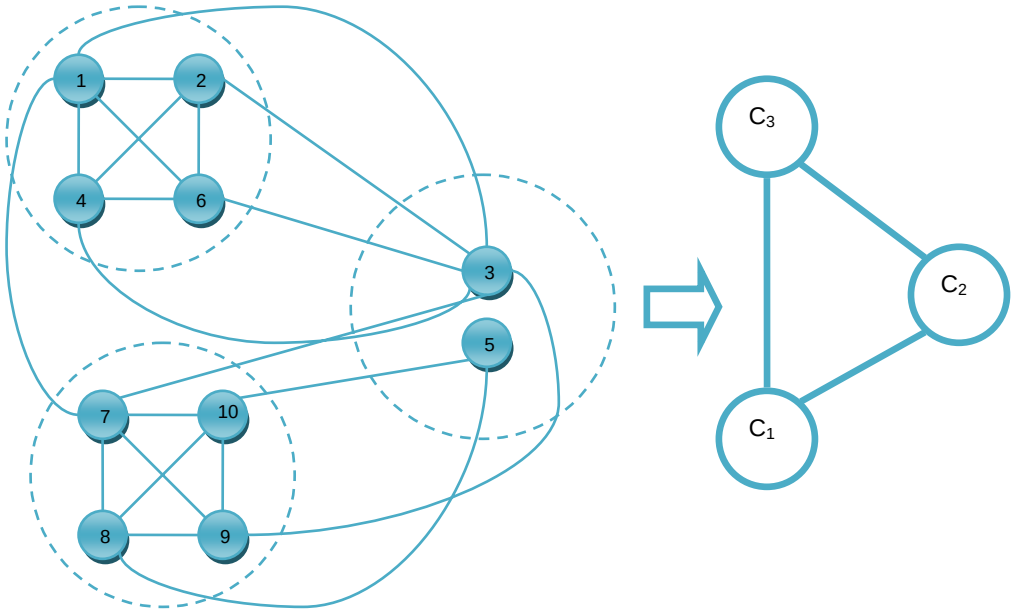


Figure 5.3 The partitioning and block diagram of graphs.

Real complexity *C(B)* is a *subjective* measure of the *system's complexity* and is relative to how one might decompose the system (Efatmaneshnik and Reidsema, 2008.a). Conversely the self complexity *C(S)* is an *objective* measure of the system and is *absolute* in being *independent* from the *type* of decomposition *P* (Efatmaneshnik and Reidsema, 2008.a). The purpose of decomposition is to reduce the initial problem complexity *C(S)* to a number of sub-problems with complexity $C(P_i)$ less than self complexity *C(S)*. The real complexity represents the *overall complexity* and the complexity of the whole system of subsystems. While being a *quality of partitioning* criteria, real complexity represents the integration effort for the whole system after

decomposition since real complexity in addition to the complexity of each sub-problem, expresses the coupling of the system of sub-systems (Efatmaneshnik and Reidsema, 2008.a). The system integration efficiency and risk is dependent on real complexity as much as it depends on self complexity (Efatmaneshnik and Reidsema, 2008.a).

Ulrich and Eppinger (2004) have argued that design efficiency can be considered directly proportional to the overall complexity of the system and decomposition affects the design efficiency. Problem decomposition must be performed very in a way that adds least possible uncertainty to the design process. Obviously by minimizing the real complexity the integration effort and risk is minimized. As such we define decomposition with minimum real complexity as immune decomposition (Efatmaneshnik and Reidsema, 2008.a).

Braha and Maimon (1998) defined design *information* as a distinct notion, and independent of its representation; information allows the designer to attain the design goals and has a goal satisfying purpose. Design can be regarded as an information process in which the information of the design increases by time: making progress and adding value (to the customer), which in a NPD system compares to producing useful information that reduces performance risk (Browning et al, 2002). Browning et al (2002) states the performance risk increases with product and problem complexity and complex system NPD involves enormous risk. Since decomposition increases the overall complexity, it must, as a result, reduce design information and increase the design risk. Figure 5.4 suggests that the process (1), by employing a better decomposition, adds less risk to the design process and reduces less information from it; and thus process (1) is likely to have a higher design efficiency (Efatmaneshnik and Reidsema, 2008.a).

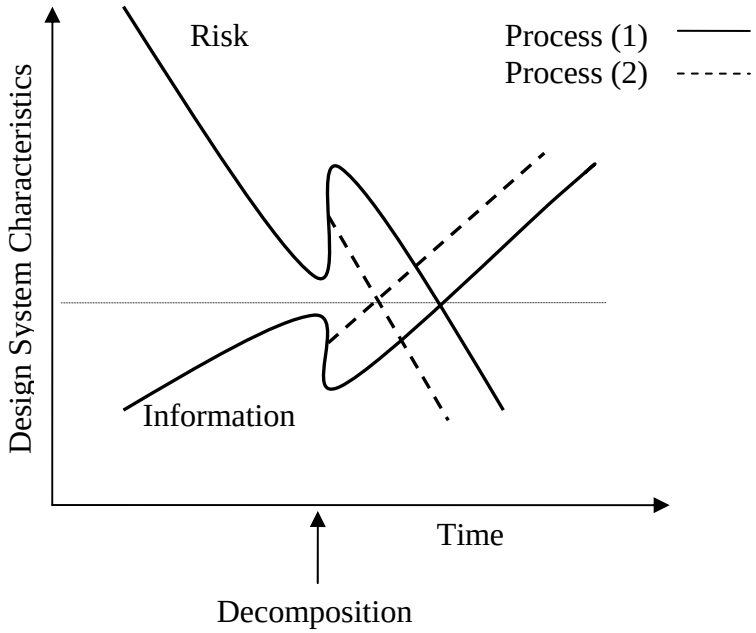


Figure 5.4 Decomposition increases risk and reduces information.

The immune decomposition is likely to lead to better, cheaper and faster production of complex products and to enhance the design process efficiency with respect to:

1. *Better products in terms of quality and robustness:* Less complexity means lower risk or rework during the design process (Browning et al, 2002) and this implies that higher quality can be achieved with less effort. Furthermore, in terms of the robustness of the product itself, immunity from a chaotic and overly sensitive response to stochastic uncertainties in the manufacturing process capabilities and during the performance cycle of the product is the direct influence of lower complexity (Efatmaneshnik and Reidsema, 2007.a).
2. *Cheaper product and process design costs:* The lower complexity structure implies less coupling between subsystems and that means less number of design iterations amongst various engineering tasks in a large problem (Smith and Eppinger, 1997) easier coordination, less conflicts arising in the integration of the product, and all these suggest a cheaper design process.

3. *Faster design process:* Designing a product is a value adding process, and that involves an accumulation of the required information to meet the design requirements (Browning et al, 2002). Obviously for less complex products this accumulation happens faster because the design process system of a less complex product would be less fragile (at lower risk) to uncertainties available in the early stages of the design process.

Figure 5.5 shows the immune decompositions of the system example in Figure 5.4, for various numbers of subsystems. The figure shows that amongst all decompositions 5-partitioning has had the minimum real complexity. The spectral diagonalization method combined with random search algorithm was used (see Section A.3.2 for the related Matlab^{TM} Codes).

Figure 5.6 compares the performance of real complexity against three other cut quality measures for *20000* randomly chosen and distinct decompositions of a randomly generated graph of order *100* and of size *150*. This figure shows that real complexity has responded differently to the number of subsystems than other measures. For this randomly chosen graph, the minimum real complexity is the global minimum (amongst different partition numbers) at bi-partitioning ($k=2$). After *6* partitions the minimum real complexity decreases by the increase of the number of subsystems. The minimum real complexity maximizes at the number of subsystems equal to *6*. Other cut quality measures show strictly decreasing linear relationship between the measure of minimums cuts and number of partitions. Another observation and interesting characteristic of real complexity is that its minimum for a particular number of subsystems maximizes at a certain number of subsystems which is different for different graphs. Amongst the quality partitioning criteria presented in Table 5.1 only modality function had this property which was always maximized at 3-partitioining.

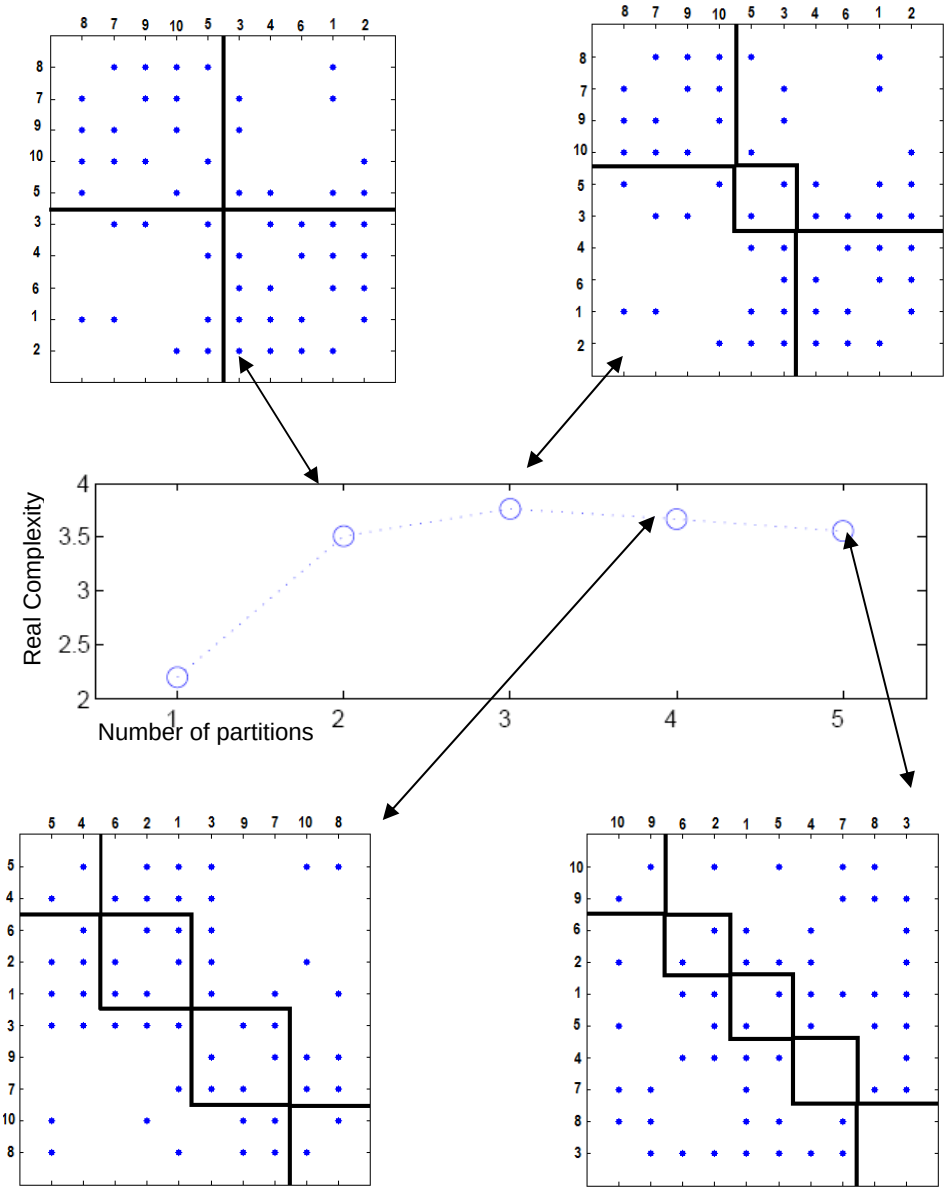


Figure 5.5 Decomposition of a DSM into various numbers of partitions.

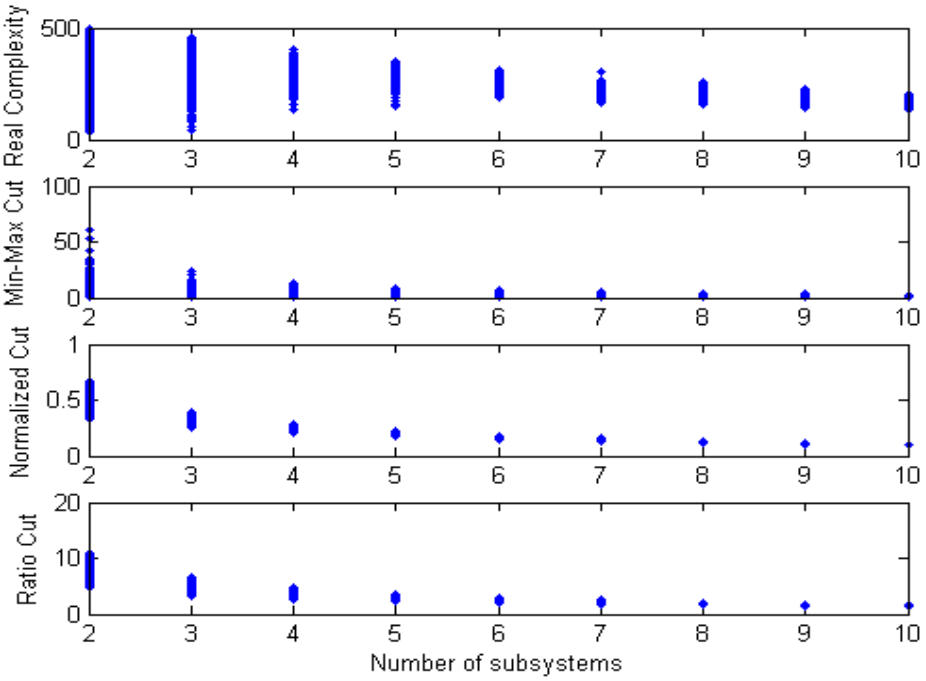


Figure 5.6 Comparison between the performance of real complexity and other cut quality measures.

5.4 Real Complexity of Overlap Decompositions

Several operation researchers have addressed the overlapping decomposition problem and have emphasized its quality improving, lead time and cost reducing benefits (Roemer and Ahmadi, 2004; Terwiesch and Loch, 1999; Krishnan et al, 1997) to the extent that design process efficiency can be regarded as proportional to the amount of the overlap between the subsystems (Clark and Fujimoto, 1989)[19]. Taking the information processing view of the product design process, Krishnan et al (1997) has argued that overlap decomposition of the problems can lead to faster information acquisition and more frequent information exchange between the subsystems enabling the concurrent execution of coupled activities in an overlapped process. However, the design teams can be collaborative only when the design space or problem space is decomposed in an overlapped manner, so

[19] As it will be clarified later in this section, this statement implies that the parts of the problem that are shared between design groups must preferably have denser connectivity (i.e. higher complexity); otherwise perfectly overlapped sub-systems lead to identical sub-problems which is equivalent to the Enlightened Engineering explained in Section 4.5.

that the design teams share some of their parameters, problems, and tasks (Efatmaneshnik and Reidsema, 2007.b).

Krishnan et al (1997) however noted that the overlap decomposition of the system must be performed very carefully as without careful management of the overlapped NPD process, the development effort and cost may increase, and product quality may worsen. It should be noted that no appropriate measure has yet been proposed to distinguish the overall behaviour of the systems under a range of possible overlap decompositions (Efatmaneshnik and Reidsema, 2007.b).

An overall complexity measure for overlap decomposition can be readily obtained by exploiting the real complexity (Efatmaneshnik and Reidsema, 2007.b). In overlap decomposition of graphs, vertices are allowed to be shared between the sub-graphs. The measurement of the overall real complexity of the overlap decompositions can be gained based on the formulation of the decentralized control strategies for overlapping information sets (Ikeda et al, 1981). Ikeda et al (1981) states that:

> The simple underlying idea is to expand the state space of the original system (design space or FL in case of the product design) so that the overlapping subsystems appear as disjoint. The expanded system contains all the necessary information about the behaviour of the original system which then can be extracted using conventional techniques devised for standard disjoint decompositions.

Figure 5.7 shows the extraction of the Complexity Based Adjacency Matrix of Block Diagram with overlapped subsystems. It can be tested that the dimension of this matrix is four whereas the number of subsystems are two.

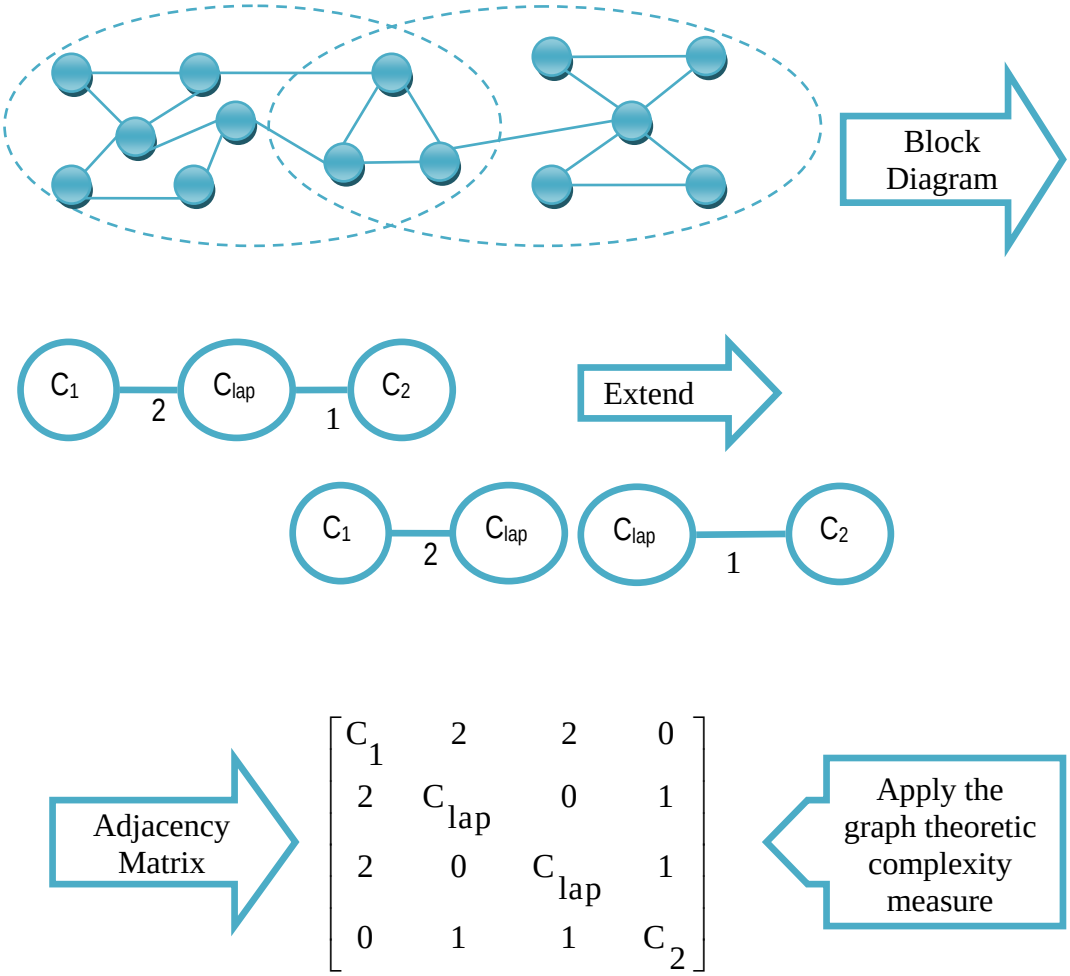


Figure 5.7 Real structural complexity measurement for overlapping subsystems. All link weights equal to one.

In general the extended Complexity Based Adjacency Matrix of Block Diagram with elements $B = [b_{ij}]$ where sub-graphs share some vertices can be defined as:

$$b_{i,j} = \begin{cases} C_i & i = j, \text{for nonoverlapping blocks} \\ C_{lap} & i = j, \text{for overlapping blocks} \\ 0 & i \neq j, \text{for overlapping blocks} \\ L_{i,j} & i \neq j, \text{for nonoverlapping blocks} \end{cases} \quad (6)$$

The effect of overlapping the decompositions on the design efficiency is a very subtle one. The integration phase of the design process is often accompanied by inadvertent information hiding due to the asynchronous information exchanges between the design teams, referred to as *design churn effect* (Eppinger et al, 2003). Design churn delays the design process convergence to a global solution.

The remedy to this effect lies in overlapping the design tasks. Overlapping leads to faster and *in time* information transfer between the design teams. This is to say that overlapping can increase the design process response and sensitivity to new information reducing design lead time and increasing design process efficiency (Figure 5.8) (Efatmaneshnik and Reidsema, 2007.b).

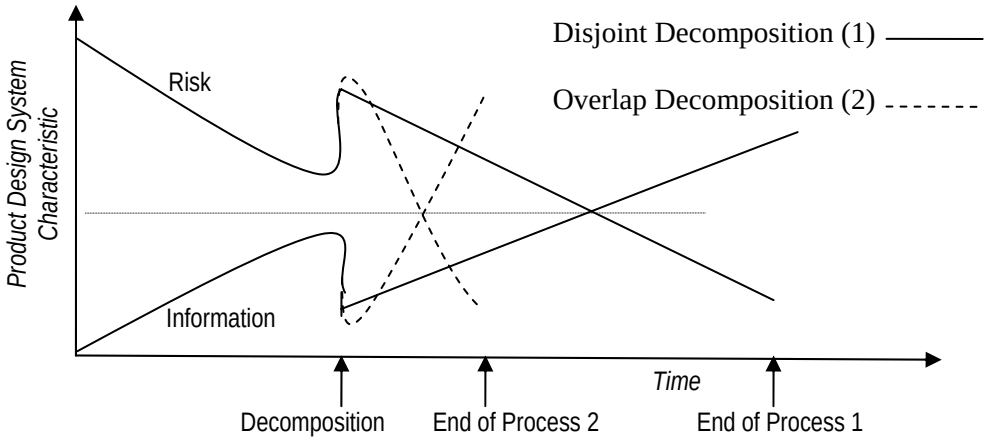


Figure 5.8 Decomposition increases risk and reduces information. Overlap decomposition makes the system to converge faster.

Overlapping leads to more real complexity in comparison with the disjoint decomposition of the design space, since overlapping virtually increases the dimensionality of the problem space (Efatmaneshnik and Reidsema, 2007.b). Thus, it is not recommended to simply overlap the subsystems as much as possible because it may lead to high overall complexity (Efatmaneshnik and Reidsema, 2007.b). We propose two seemingly conflicting objectives when overlapping subsystems (Efatmaneshnik and Reidsema, 2007.b):

1. To minimize the real complexity of the whole (extracting immune decompositions).
2. To maximize the sum complexity of the overlapped parts. The complexity sum of the overlapped parts (C_{lap}) is representative of how much the system is overlapped

Figure 5.9 demonstrates the real complexity and the overlapping degree (complexity sum of the overlapping parts) of many random decompositions (Efatmaneshnik and Reidsema, 2007.b). This figure shows the desired region for the decompositions, the characteristic of which is minimum overall real complexity (maximum efficiency, minimum fragility of the corresponding design process or immunity) and maximum overlapping complexity (high sensitivity of the design process to new information) (Efatmaneshnik and Reidsema, 2007.b).

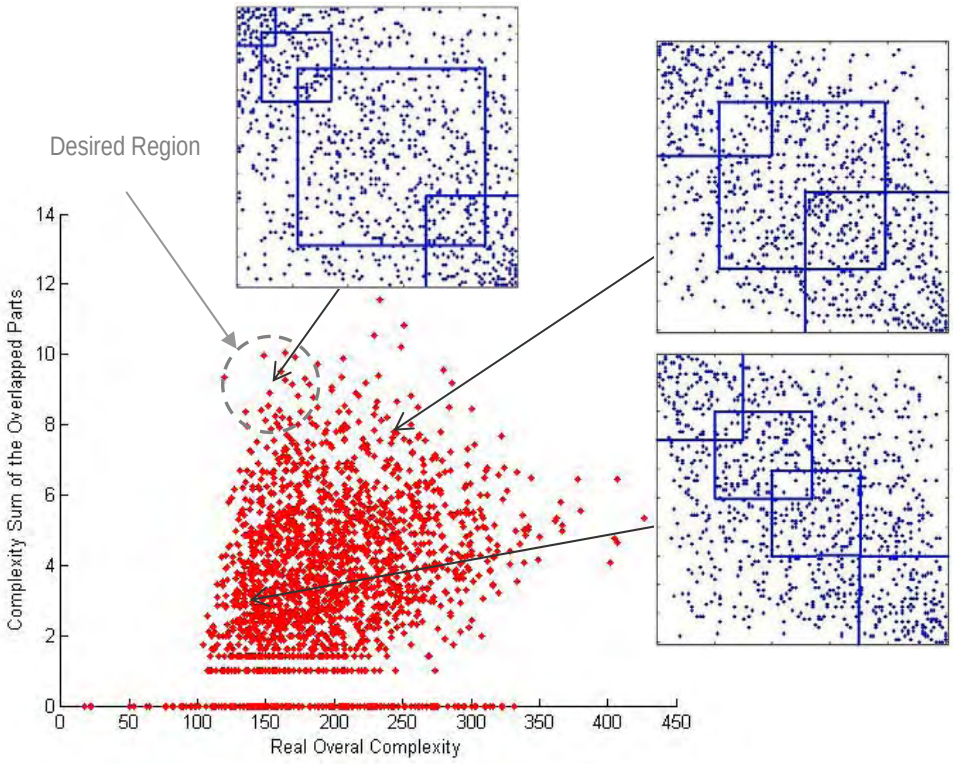


Figure 5.9 Extracting the desired overlap decompositions for routine design process of complex systems by random search and spectral diagonalization.

5.5 On Decomposability

According to Edmonds (1999) the general area of decomposability is covered by Arbib (1974), Conant (1972), Dussauchoy (1982), Naylor (1981) and Steel (1992). The complexity based approach to decomposability has been considered by Edmonds (1999). He used the "analytical complexity measure" to determine the decomposability of the syntactical expressions in formal languages. The ease with which a system can be decomposed into sub-systems has close connections with the real complexity. Real complexity provides a medium for decomposability testing. A system that is not decomposable is irreducible. Before proceeding lets introduce the lower bound for the real complexity that will be used in the subsequent discussion.

Observation: Given system S with graph G_S as its graphical representation and for all *k*-partitioning $P = \{P_1,P_2,\ldots,P_k\}$ of *S* with $i=\{1\ldots k\}$,the following is valid (see Section A.1 for proof):

$$C(B) \geq C(S) \geq C(P_i) \qquad (7)$$

Where *B* is the Complexity Based Adjacency Matrix of the Block Diagram of decomposition *P* on G_S. This means that the lower bounds for real complexity are the self complexity, and also the complexity of each of the sub-systems (Efatmaneshnik and Reidsema, 2008.a). The equality happens when the system can be fully decomposed. This observation is important since it indicates that decomposition cannot decrease the overall complexity of a problem, and the perceived complexity after decomposition: *C(B)* is never less than the complexity before decomposition *C(S)*. Note that in the block diagram the information that indicates which nodes in different subsystems have been linked is lost (Efatmaneshnik and Reidsema, 2008.a). Similarly when a system is decomposed the information indicating which vertices are linked to which ones in other subsystems is also lost. Klir (2003) states:

> *When* a system is simplified it is unavoidable to lose some of the information contained in the system; the amount of

information lost results in the increase of an equal amount of relevant uncertainty. Any kind of simplification including break of the overall system to subsystems can increase uncertainty.

More uncertainty implies more complexity and thus the lower bound for real complexity *must* be and cannot be anything but the complexity of the system before decomposition (self complexity) (Efatmaneshnik and Reidsema, 2008.a); in other words *facing more overall complexity as the price for the tractability of sub-problems* can, in fact, be regarded as a read of *no free lunch theorem* (Efatmaneshnik and Reidsema, 2008.a).

Considering decomposition $P=\{P_1, P_2 ,\ldots,P_k\}$ of system *S*. Then complexity of *the whole* for the system can be represented by the real complexity *C(B)* and that of *the parts* by the complexity of individual subsystems $C(P_i)$. Real complexity can then explain whether *the whole is more than sum of the parts* (Efatmaneshnik and Reidsema, 2008.a):

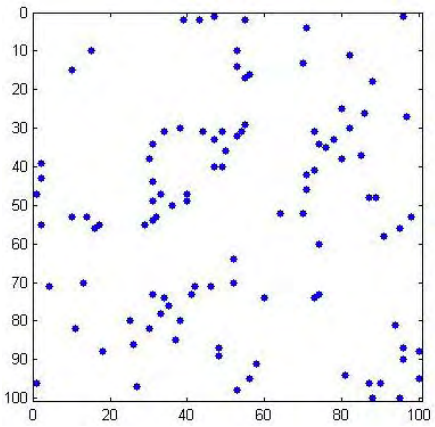
$$C(B) \overset{?}{\geq} \sum_{i=1}^{k} C(P_i) \qquad (8)$$

When and if (8) holds, the system cannot be or has not been reduced to the sum of its constituents and therefore the system is irreducible (Efatmaneshnik and Reidsema, 2008.a). It should be reminded that the graph theoretic complexity measure is a measure of the intensity of emergence and possibility of emergent surprise (characteristics). Therefore, the integration of a system, being decomposed in a way that (8) holds, is prone to a significant amount of risk; it is very likely that the whole system of sub-systems shows emergent properties that do not exist in the sub-problems (Efatmaneshnik and Reidsema, 2008.a). The integration, in such conditions cannot be implicitly performed while the sub-problems are being solved concurrently because as much attention must be given to communication between the design teams (Efatmaneshnik and Reidsema, 2008.a). Where all viable decompositions of a system have the property of (8), then decomposition is not a valid and robust methodology for problem solving

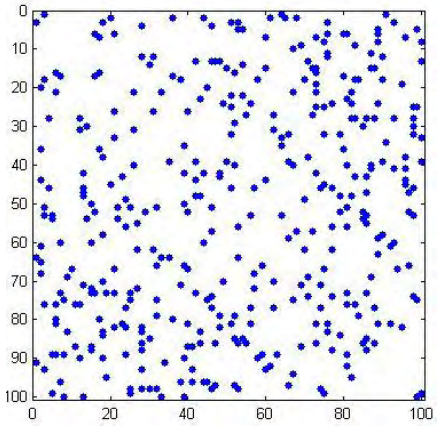
(however it may not be impossible). In such cases the whole is more than sum of the parts (regardless of what is deemed as parts), and distributed problem solving must be forsaken (Efatmaneshnik and Reidsema, 2008.a); the problem may be tackled satisfactorily as a whole utilizing several design teams working and competing parallel to each other (with no collaboration).

When the opposite of (8) is true and a decomposition *P* can be found in a way that the real complexity is less than sum of the subsystems complexities then the system is reducible. Under such circumstances, the complexity of the whole *can be* reduced to the complexity of the individual components. Obviously the reducibility depends on two main factors: the self complexity and decomposition restriction.

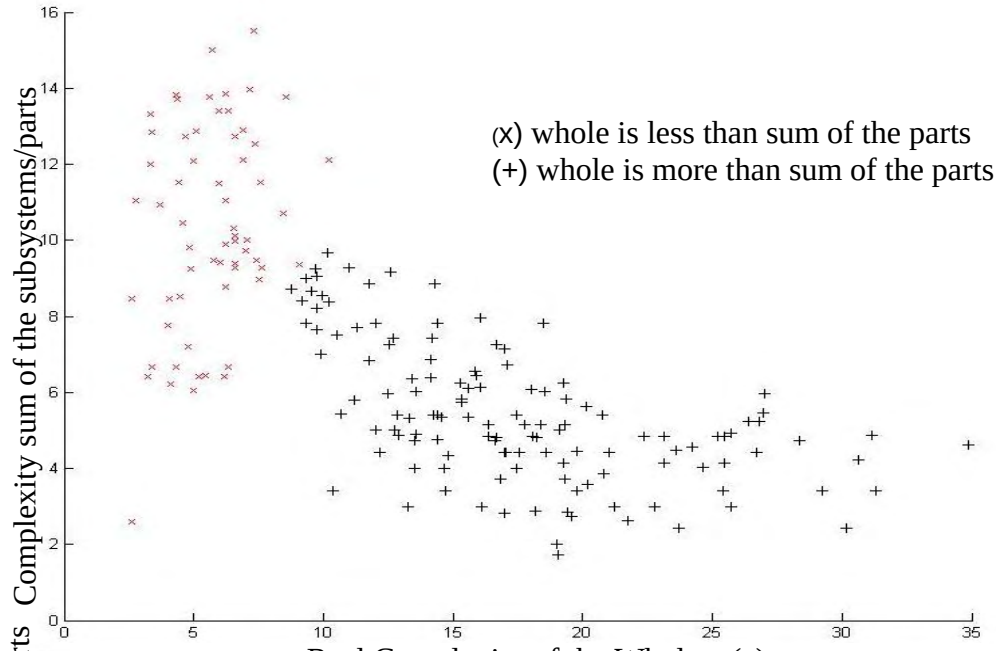
In Figure 5.10 two systems are decomposed in many different ways (different subsystems). The two left matrices are the PDSM (or self) of systems S_a and S_b (each dot represents a link between the two nodes or variables). The system S_a in this figure is of order *100* and size *100* whereas system S_b is of the same order and size *400*; system S_b is four times denser that S_a. Some decompositions of the system S_a do not render the whole (real complexity) as being more than the sum complexity of the parts (Figure 5.10(a)). Thus system S_a can be reduced to the sum of its parts by some decomposition(s). On the contrary regardless of how system S_b may be decomposed, it cannot be reduced to the sum of its parts, and therefore system S_b is irreducible (Figure 5.10(b)). Figure 5.11 compares the whole and sum of the parts for many decompositions of the system presented in Table 5.1. This system was decomposed in *200* different ways all of which had relatively balanced subsystems (equally ordered). The system was irreducible under almost all the viable decompositions.



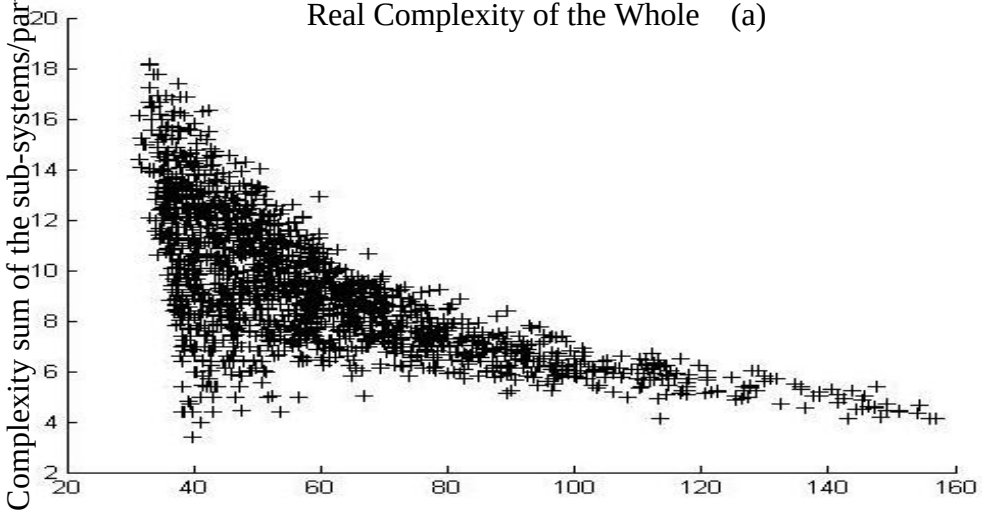
S_a



S_b



(a)



Real Complexity of the Whole (b)

Figure 5.10 Using real complexity to test decomposability.

For irreducible systems decomposition does not lead to higher tractability. Decomposition for such systems can further complicate the process of collaboration between design teams: the overall effort of integrating the system of subsystems amounts to more than sum of the efforts spent on integrating each subsystem. This notion indicates substantial amount of rework and design iteration in the design of complex systems by means of decomposition. As such the application of concurrent engineering for some highly complex systems can be questioned.

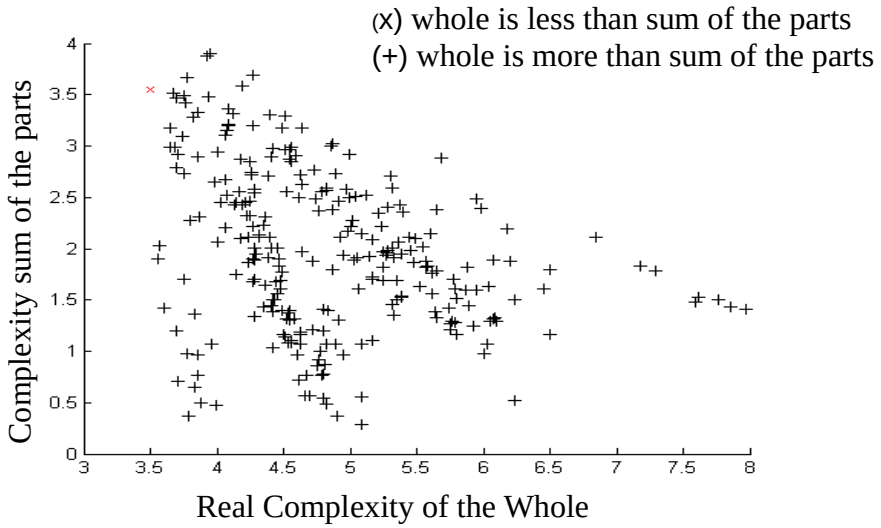


Figure 5.11 For almost all decompositions of the example PDSM (except one) the whole is more than sum of the parts.

6 Integration and Organizational Immunity

In Chapter 4 we argued that the organisational DSM must be derived directly from the simulated PDSM, and that a direct mapping can be used to force the organisation structure to mirror the product architecture. This allows for predefined communication and information exchange channels, in a large and complex environment, which, per se, prevent the process from spiralling out of control (Mihm and Loch, 2006). For example, consider the example PDSM introduced in Table 4.1 being decomposed as shown in Table 6.1. The design of each subsystem is then assigned to a design team. From the PDSM a team based DSM (Table 6.2) is derived by summing up the amount of information exchange (dependency) of the design variables each team is responsible for. This predicted team based matrix reflects the *likely* information exchanges as well as the internal complexity of the problems that the design teams ought to deal with (Efatmaneshnik and Reidsema, 2009).

Table 6.1 The variables of Table 4.1 are rearranged to form three subsystems.

		Subsystem1			Subsystem2				Subsystem3		
		V5	V4	V2	V10	V8	V7	V9	V6	V1	V3
Subsystem1	V5	0	0.45	0.44	0	0	0.02	0.02	0.53	0.22	0.11
	V4	0.34	0	0.65	0	0	0	0	0.43	0.16	0.64
	V2	0.3	0.12	0	0	0	0.2	0.1	0.2	0.76	0.12
Subsystem2	V10	0.01	0	0.18	0	0.8	0.1	0.9	0	0	0
	V8	0.01	0	0	0.1	0	0.2	0.4	0	0.01	0
	V7	0	0	0	0.3	0.45	0	0.1	0	0.12	0.02
	V9	0	0	0	0.5	0.2	0.7	0	0	0	0.15
Subsystem3	V6	0	0.34	0.78	0	0	0	0	0	0.77	0.31
	V1	0	0	0.53	0	0	0	0	0.1	0	0.32
	V3	0	0	0.11	0.72	0	0.3	0.52	0.2	0.45	0

While this approach is necessary as it yields a predefined map for problem solving it is not sufficient since it does not allow for innovation (Efatmaneshnik

and Reidsema, 2009). Innovation is often regarded as a means of achieving competitive edge over other NPD companies. However, in the case of complex systems innovation has a more vital role to play: integration (Efatmaneshnik and Reidsema, 2009). Before proceeding let's consider a detailed account of the innovation types.

Table 6.2 The predicted team based DSM for the entire system.

-	Team1	Team2	Team3
Team1	C_{T1}	0.34	3.17
Team2	0.2	C_{T2}	0.3
Team3	1.76	1.52	C_{T3}

6.1 Radical Innovation

Henderson and Clark (1990) demonstrated that there are different kinds of innovation as depicted in Figure 6.1 where innovation is classified along two dimensions; the horizontal dimension captures an innovation's impact on components (subsystems), while the vertical dimension captures its impact on the linkages between core concepts and components (Henderson and Clark, 1990). Incremental innovation refines and extends an established design. Improvement occurs in individual components, but the underlying core design concepts, and the links between them, remain the same. Modular innovation on the other hand, changes only the core design concepts without changing the product's architecture. Architectural innovation changes only the relationships between modules but leaves the components, and the core design concepts that they embody, unchanged. Radical innovation establishes a new dominant design, hence a new set of core design concepts embodied in subsystems that are linked together in a new architecture. We can say that radical innovation embodies both modular and architectural innovation.

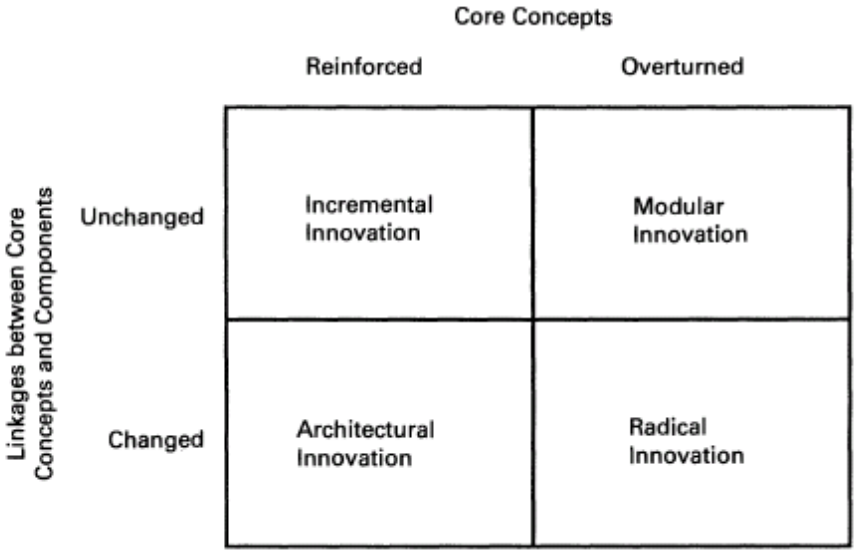


Figure 6.1 Different types of innovation, after Henderson and Clark (1990).

An organization's communication channels, both formal and informal are critical to achieving radical and architectural innovation (Henderson and Clark, 1990). The communication channels that are created between these groups will reflect the organization's knowledge of the critical interactions between product modules. An organization's communication channels will embody its architectural knowledge of the linkages between components that are critical to effective design (Henderson and Clark, 1990). They are the relationships around which the organization builds architectural knowledge.

Innovation processes in complex products and systems differ from those commonly found in mass produced goods (Hobday et al, 2000). The creation of complex products and systems often involves radical innovation, not only because they embody a wide variety of distinctive components and subsystems (modular innovation), skills, and knowledge inputs, but also because large numbers of different organizational units have to work together in a collaborative manner (architectural innovation). Here, the key capabilities are systems design, project management, systems engineering and integration (Hobday et al, 2000).

Integration in complex system and product design is aimed at making the solutions to sub-problems compatible with each other and is possible through

innovation (Efatmaneshnik and Reidsema, 2008.b). The innovation that integrates the complex system must be radical innovation and creativity (Sosa and Gero, 2005) and is an emergent property of the entire system rather than the property of the sub-solutions to the individual sub-problems (Sosa and Gero, 2004). A property that is only implicit, i.e. not represented explicitly, is said to be an emergent property if it can be made explicit and it is considered to play an important role in the introduction of new schemas (Gero, 1996). The radical innovation and coherency in an engineered large scale system is emergent and obtained in a self organizing fashion in a multi agent environment.

When designing self-organizing multi-agent systems with emergent properties, a fundamental engineering issue is to achieve a macroscopic behaviour that meets the requirements and emerges only from the behaviour of locally interacting agents (Efatmaneshnik and Reidsema, 2008.b). Agent-oriented methodologies today are mainly focused on engineering the microscopic issues, i.e. the agents, their rules, how they interact, etc, without explicit support for engineering the required macroscopic behaviour. As a consequence, the macroscopic behaviour is achieved in an ad-hoc manner (Wolf and Holvoet, 2005).

Creativity requires ad hoc communication in which the need to communicate often arises in an unplanned fashion, and is affected by the autonomy of the agents to develop their own communication patterns (Leenders et al, 2003). It is thus obvious that, a fixed organizational structure with established patterns of communication is unlikely to deliver new complex structures (products) (Efatmaneshnik and Reidsema, 2008.b). As such we propose that the derived team based DSM must remain as a backbone and suggestion for intra team communication (Efatmaneshnik and Reidsema, 2008.b). Leenders et al (2003) also showed that team creative performance will be negatively related to the presence of central team members (including brokers, mediators and facilitators) in the intra-team communication network. However a great threat to multi agent design systems is chaos. To immunise the design organization from this threat we suggest holistic process monitoring described next.

6.2 Holistic Process Monitoring

When the inherent nature of a complex task is too large, a better solution is to create an environment in which continuous innovation can occur (Bar-Yam, 2004). This can be accomplished through process monitoring: Bayrak and Tanik (1997) reported that improving the design process, which increases the product quality without increasing the design resources, is possible by providing feedback to the designer to help him/her understand the nature of the design process. Therefore, the nature of the design becomes easier to analyse if there are metrics obtained from activity monitoring (Bayrak and Tanik, 1997).

Since the design process of the complex systems by concurrent engineering is an emergent process (Cisse et al, 1996), holistic metrics are required to monitor the design process. One such metric is the cognitive complexity of a process that is defined as the ability of a problem solver to flexibly adapt to a multidimensional problem space (Lee and Truex, 2000). Cognitive complexity represents the degree to which a potentially multidimensional cognitive space is differentiated and integrated (Lee and Truex, 2000). A problem solver (a person, organization or a multi agent system) with higher cognitive complexity is more capable of having creative (and holistically correct) outcomes.

We suggest measuring the cognitive complexity of a multi agent design process as a function of the amount of information exchange between the design agents (Efatmaneshnik and Reidsema, 2009). It is assumed here that all the design agents are equal in their cognitive complexity and problem solving abilities but the violation of this assumption does not jeopardize the method (the agents can be ranked by their cognitive complexity). The required cognitive complexity is, thus, derived by applying the graph theoretic complexity measure (presented in the Appendix) to the team based DSM at all instances of design process (Efatmaneshnik and Reidsema, 2009).

Cognitive complexity is an emergent property of the design system. As such, cognitive complexity must be measured, for various hierarchies that the

design system might have, as the *real cognitive complexity*[20] of that level. This can be derived from the fact that information exchange and communication between one element of a connected subset and that of a different subset will indirectly affect the entire elements of both subsets. In the context of the design systems, this statement can be reiterated: different design teams (collection of design agents) and various design systems (groups of teams) deal with each others' emergent properties (real cognitive complexity) rather than the characteristics of their individual elements, however, the actual communication might occur in between the design agents (elements of the teams).

Figure 6.2 shows a hierarchical design system with three hierarchical levels: design teams, design systems and the design system of systems. CC_Ts are the emergent cognitive complexity of the design teams, CC_Ss are the emergent cognitive complexity of the design systems (the details of one of which is only shown) and CC_G is the cognitive complexity of the entire design system of systems. It is important to note that cognitive complexity could otherwise be measured for a flat picture of the system. We will, however, use these hierarchical notions in the next Chapter where the design systems can work in parallel on various abstraction levels of the problem, where each system may contain several design teams.

The main complication here is the way in which the information exchange is measured. Kan and Gero (2005) suggested the use of entropy based measures for evaluation of information content of a design agent's interactions. We suggest using a fuzzy method by simply asking the design participants to tag qualitative and quantitative information content of their interactions with a single fuzzy variable, e.g. high, low, and medium, etc. The fuzzy interactions tags can then be defuzzified, which is the process of producing a quantifiable result in fuzzy logic, according to a simple fuzzification rule such as that in Figure 6.3. Note that in this figure, depending on the hierarchical level, N is the maximum amount of information exchange between the collective entities of the hierarchy (and not just the elements). Also note that the fuzzification scheme in this figure is the simplest

[20] This is similar to the notion of real complexity.

possible scheme where all the membership functions are equal to one; thus this scheme actually depicts crisp boundaries between the sets. Obviously elaborate fuzzy membership functions such as the trapezoid and triangular can increase the precision of defuzzification.

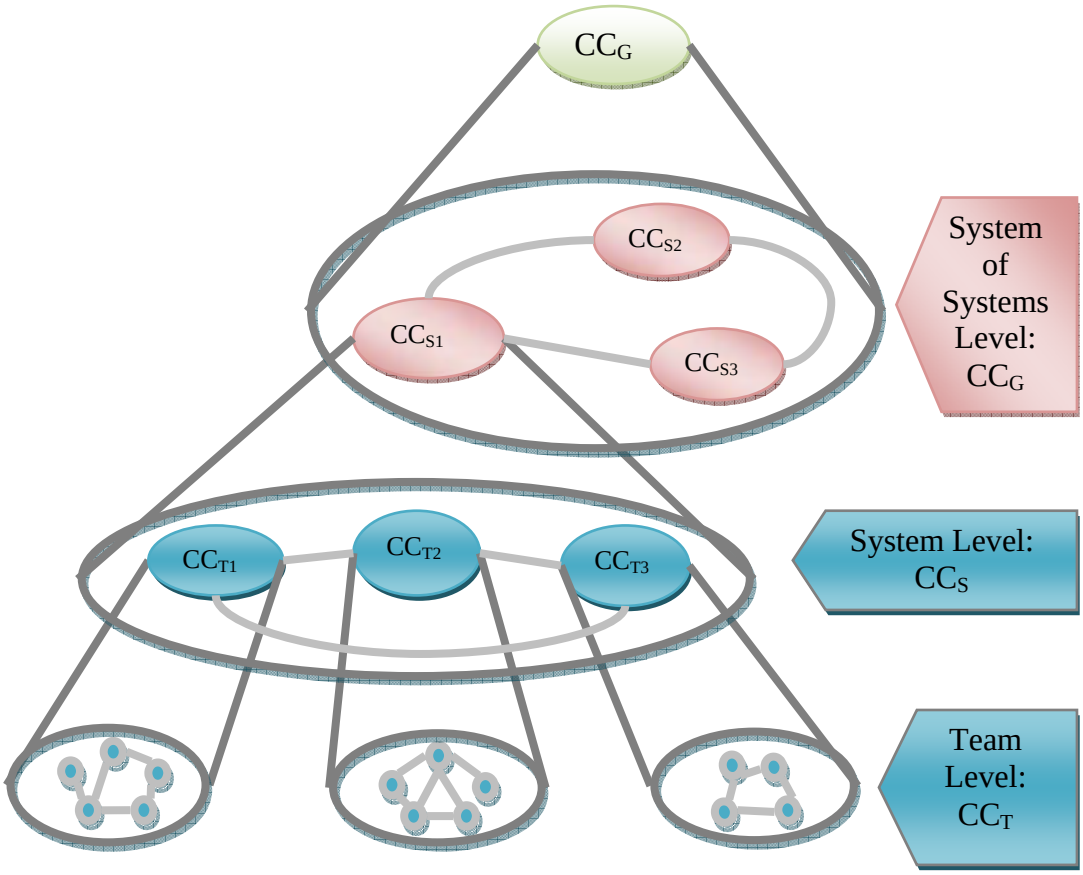


Figure 6.2 Shows the emergence of cognitive complexity at three hierarchical levels.

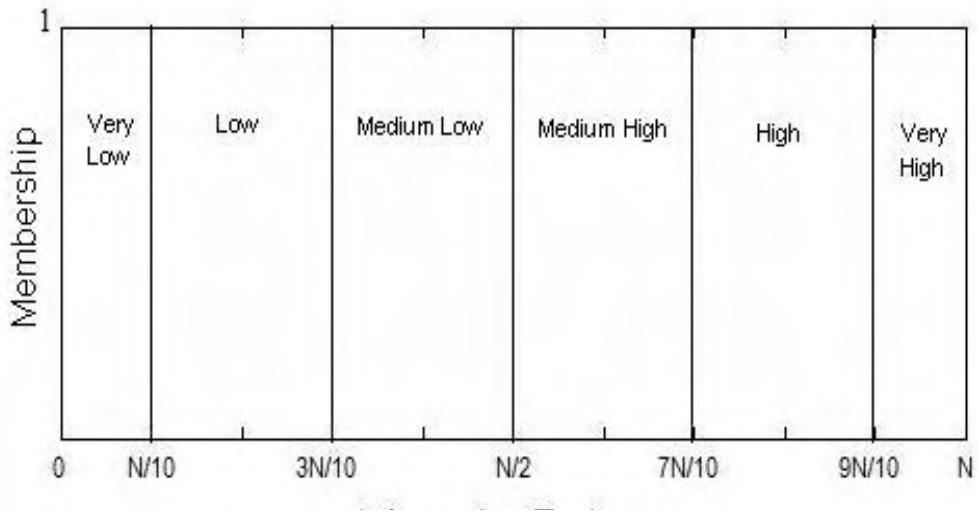


Figure 6.3 A simple fuzzification scheme.

The example in Figure 6.4 further illustrates the process of defuzzification. This figure shows an example of the *observed* fuzzy team based DSM (Figure 6.4(a)) in charge of designing *subsystem 2* in Table 6.1. Four design agents are assumed to be in *team 1*, the internal interactions of which create the cognitive complexity of *team 1* (CC_{T1}). This DSM is then defuzzified based on the information in Figure 6.3 and that the maximum amount of information exchange for that *subsystem 2* was *0.9*. (Figure 6.4(b)). The procedure of estimating the cognitive process complexity for a design system comprised of three design teams is illustrated in Figure 6.4(c) and Figure 6.4(d).

Having measured the cognitive complexity of all the design teams (CC_{T1}, $CC_{T2,}$ CC_{T3}) at any instance, and knowing the amount of intra team information exchange, one can measure the cognitive complexity of the entire system (CC_S). The *N* values in this stage are based on the maximum information exchanges in Table 6.2. Similarly, the cognitive complexity of the system of systems, at any design instance, can be estimated. By means of these fuzzy transformations we can compare the complexity of the design problem solving system with that of design problem at instance of the problem solving process. Note that the maximum and minimum complexity values for the design system of teams and the design system of systems must reflect the *real* minimum and maximum complexities corresponding to those hierarchical levels.

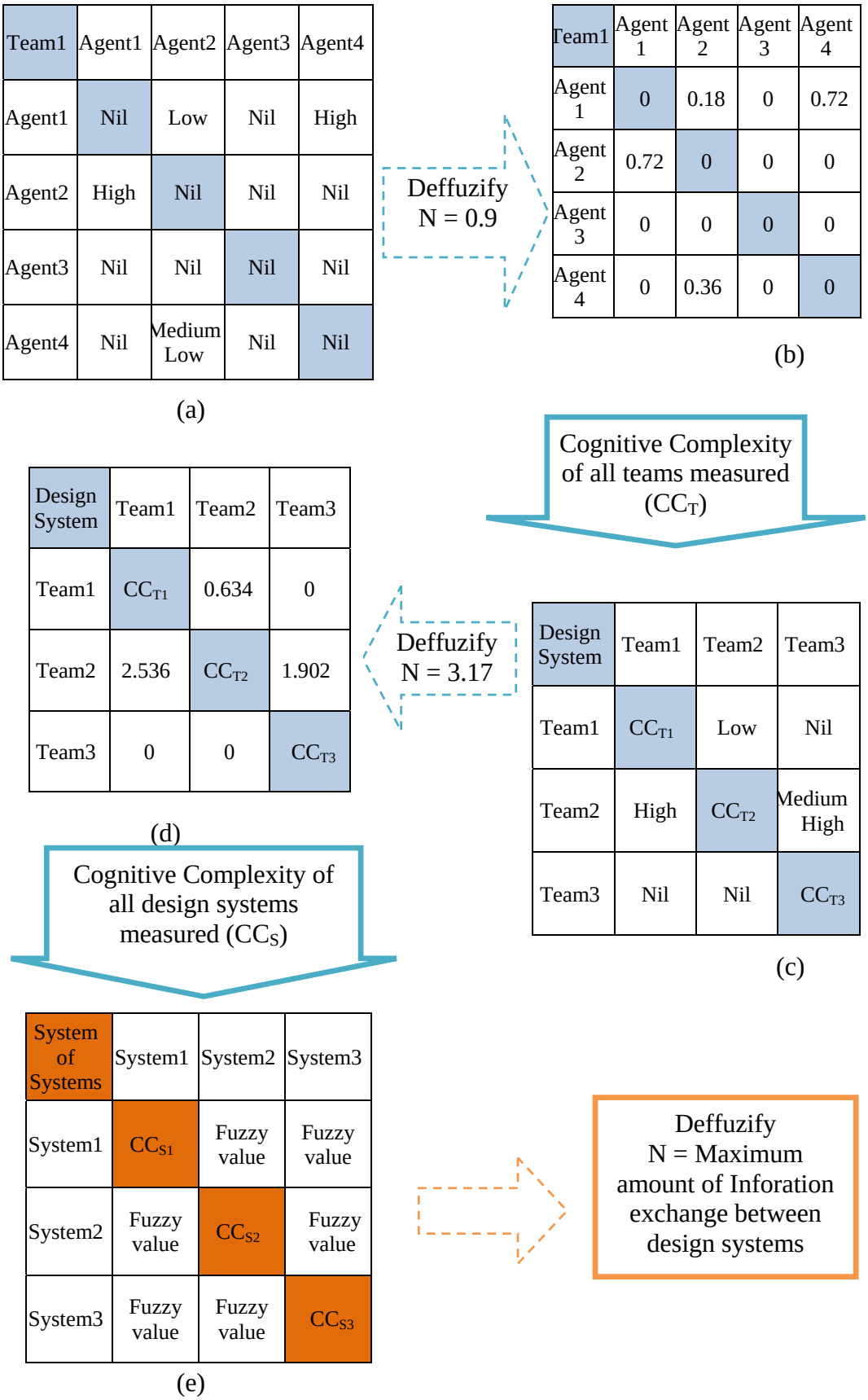


Figure 6.4 Measuring the cognitive complexity of the design process at certain design instance.

6.3 An Artificial Immune Algorithm

We argue that the Cognitive Complexity of the design system must be lower and upper bounded by the minimum and maximum complexity of the problem. This constitutes the proposed method of this thesis for immunization of the design organization (Efatmaneshnik and Reidsema, 2008.b). Two premises form the basis of this argument:

1. In order to solve a problem, the problem solver needs to have a (cognitive) complexity more than or equal to the problem complexity (Bar-Yam, 2004). This is indeed a specific interpretation of the law of requisite variety in Cybernetics (Ashby, 1970). As such the cognitive complexity of the process must be more than the minimum complexity of the problem.
2. The tendency of the design agents for increased collaboration by means of information exchange do not necessarily lead to more overall cognitive complexity of the design system. By testing a sample of 44 NPD organizations, Leenders et al (2003) have shown that the performance of innovation networks (innovation teams) has an inversely U-shape relationship to frequency of intra team cooperation.

Balanced participation of design players in a design decision making process is favoured against increasing information flow between the design players to a maximum (Chiva-Gomez, 2004). This is to say unnecessary information exchange may lower the overall cognitive complexity. It is obvious that the cognitive complexity of the process need not be more than the real maximum complexity of the problem. In fact more than the required information exchange can lead to creativity blocks (Leenders et al, 2003) which can be termed a chaotic situation. Thus an upper bound for the cognitive complexity of the process is the maximum complexity of the problem (Efatmaneshnik and Reidsema, 2009):

$$C_{max} \geq C_C \geq C_{min} \quad (1)$$

According to Stacey (1995) innovation in a multi agent environment is the result of communication between social agents that happens in a self organizing fashion and when the multi agent system finds itself on the so-called edge of chaos. When the cognitive complexity of the process is more than the minimum complexity and tending towards the maximum complexity of the problem, the design system might be on the edge of chaos but certainly not chaotic (Efatmaneshnik and Reidsema, 2008.b). Besides for collaborative multi agent systems with cognitive complexity less than the minimum complexity, the design process is certainly away from the edge of chaos, thus the design system does not have sufficient functionality to deliver radical innovation in an optimal and efficient manner (Efatmaneshnik and Reidsema, 2008.b). For systems that are excessively persistent in collaboration and cooperation, the cognitive complexity may reduce and chaos appears in such condition (Efatmaneshnik and Reidsema, 2008.b). Chaos makes the design process fragile and susceptible to failure and reduces the design system efficiency (Efatmaneshnik and Reidsema, 2008.b).

Figure 6.5 shows that the design system's overall cognitive complexity increases only to a certain threshold by the increase in the tendency of the design agents for exchanging design information. In order to ensure the health of the design process it is necessary to ensure that the overall cognitive complexity stays away from the maximum complexity and above the minimum. This way the real minimum and maximum complexity that are obtained using the initial Monte Carlo Simulation of the complex product (low level product knowledge) are used to monitor the efficiency and effectiveness (health) of the complex product design process. This may be achieved through monitoring the design process. The process monitoring here serves the purposes of meeting the design objectives (quality, cost, and lead time) by immunizing the design system against chaos and lack of effectiveness. This immunization enables the design system to integrate the complex system and product through emergence of radical innovation.

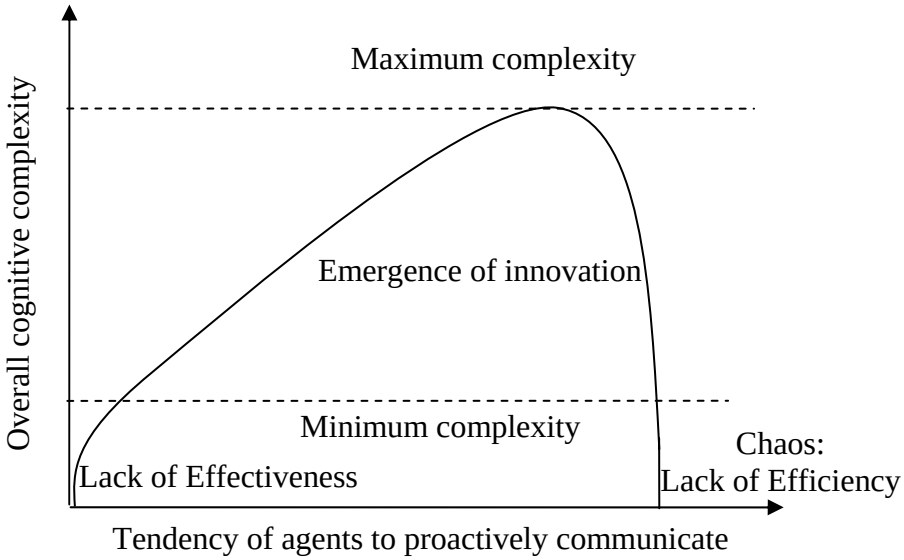


Figure 6.5 Design process functionality versus process complexity.

This is indeed an immune algorithm that would allow for the emergence of innovation. According to Cohen (2007) the immune system is a ‘‘computational strategy’’ to carry out the functions of protecting and maintaining the body. Cohen’s maintenance role of the immune system requires it to provide three properties:

1. *Recognition*: to determine what is right and wrong.
2. *Cognition*: to interpret the input signals, evaluate them, and make decisions.
3. *Action*: to carry out the decisions.

These properties are provided via a cognitive strategy in which self-organization of the immune system is used to make decisions (Timmis et al, 2008). The stages correspond to the holistic control of the system, which is to immunize or ensure the realization of self-organization, by using a complexity measure:

1. *Recognition*: recognizing the lower and upper complexity bounds.
2. *Cognition*: to evaluate the instantaneous complexity of the system.
3. *Action*: to maintain this complexity in between the bounds at all times.

An immune algorithm is a plan that determines how the components of the systems are going to interact to determine the system dynamics (Timmis et al, 2008). For example Dasgupta (1998) examined various response and recognition mechanisms of immune systems and suggested their usefulness in the development of massively parallel adaptive decision support systems. Lau and Wong (2004) presented a multi agent system that could imitate the properties and mechanisms of the human immune system. The agents in this artificial immune system could manipulate their capabilities to determine the appropriate response to various problems. Through this response manipulation, a non-deterministic and fully distributed system with agents that were able to adapt and accommodate to dynamic environment by independent decision-making and inter-agent communication was achieved (Lau and Wong, 2004). Ghanea-Hercock (2007) maintained a multi agent simulation model that could demonstrate self organizing group formation capability and collective immune response. He showed that the network of agents could survive in the face of continuous perturbations. Fyfe and Jain (2006) presented a multi agent environment in which the agents could manipulate their intentions by using concepts suggested by artificial immunes system to dynamically respond to challenges posed by the environment. Goel and Gangolly (2007) presented a decision support for robust distributed systems security based on biological and immunological mechanism.

We define a system to be immune to chaos and preserving its holistic self characteristics if its complexity is in between the minimum and maximum complexity bounds (Efatmaneshnik and Reidsema, 2008.b). The proposed immune algorithm provides a collective immune response for engineering design of complex systems and is illustrated in Figure 6.6. This figure shows the proposed algorithm for a system with only one hierarchical level. As shown in this figure upon the arrival of new information (finalization of the values of one or some of the design variables), it can be incorporated into the design system by performing a new simulation and measuring the values of the minimum and maximum complexity. This way the bounds reflect the true status of the low level design problem on the design system.

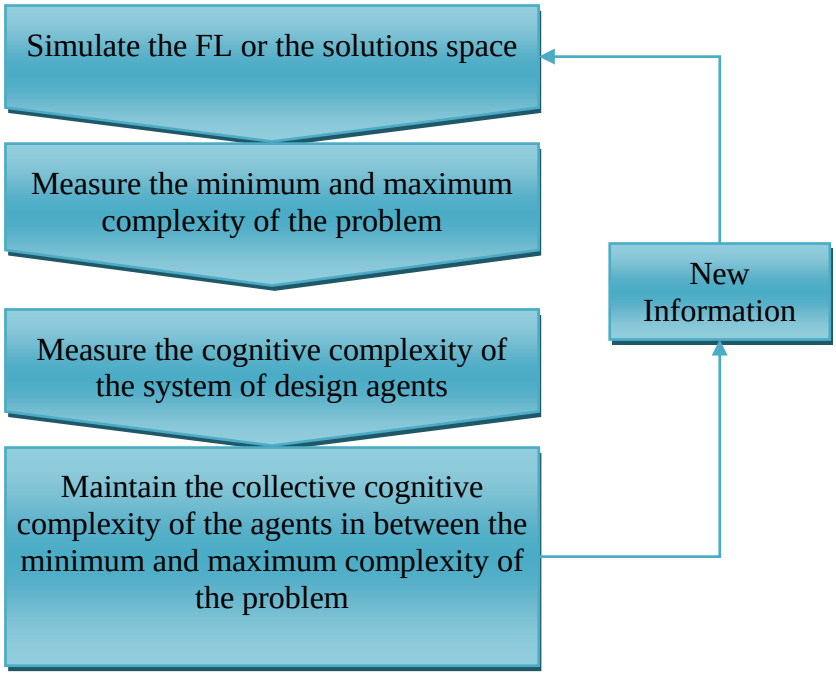


Figure 6.6 An Immune algorithm for design of complex systems.

For an example and more clarification consider the problem with simulated PDSM that was presented in Table 4.1. From this table and its corresponding hypergraph in Figure 3.11 the following prosperities can be determined:

$$C_{min}=1.64 \quad (2)$$

$$C_{max} = 4.19 \quad (3)$$

$$N=0.9 \quad (4)$$

Where N is the maximum amount of information exchange, C_{min} is the minimum complexity and C_{max} is the maximum complexity. Each of the variables in Table 4.1 is assigned to a design agent to determine its value. The design agents are, then, asked to report the amount of their information exchanges and interactions with each other. Consider Table 6.3 as the reported or observed fuzzy team based DSM at a given instance of the design process.

Table 6.3 The monitored (reported) fuzzy team based DSM

-	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
A1	-	Low	-	Low	Low	High	Low	VL	-	-
A2	H	-	L	MH	ML	H	-	-	-	L
A3	ML	L	-	MH	L	ML	VL	-	L	-
A4	L	MH	MH	-	ML	L	-	-	-	-
A5	L	ML	L	ML	-	-	-	VL	-	VL
A6	H	H	ML	ML	-	-	-	-	-	-
A7	L	-	VL	-	-	-	-	VH	VL	L
A8	VL	-	-	-	VL	-	L	-	L	H
A9	-	-	L	-	-	-	H	L	-	H
A10	-	L	-	-	VL	-	L	H	H	-

Table 6.4 is resulted by defuzzifying the observed team based DSM according to the fuzzy rule in Figure 6.3. The complexity of the defuzzified team based DSM is *1.19* which is in between the minimum and maximum complexity bounds (*1.64*, and *4.19*).

$$C_C=1.9191 \quad (5)$$

This information indicates that that design agents can safely communicate more actively to increase the cognitive complexity of the system.

Table 6.4 The defuzzified monitored team based DSM

-	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10
A1	0	0.18	0	0.18	0.18	0.72	0.18	0.04	0	0
A2	0.72	0	0.18	0.56	0.36	0.72	0	0	0	0.18
A3	0.36	0.18	0	0.56	0.18	0.36	0.04	0	0.18	0
A4	0.18	0.56	0.56	0	0.36	0.18	0	0	0	0
A5	0.18	0.36	0.18	0.36	0	0	0	0.04	0	0.04
A6	0.72	0.72	0.36	0.36	0	0	0	0	0	0
A7	0.18	0	0.04	0	0	0	0	0.9	0.04	0.18
A8	0.04	0	0	0	0.04	0	0.18	0	0.18	0.72
A9	0	0	0.18	0	0	0	0.72	0.18	0	0.72
A10	0	0.18	0	0	0.04	0	0.18	0.72	0.72	0

One last point here is that, in cases where the design system is hierarchical, the minimum and maximum complexity bounds must be measured from the corresponding Complexity based Adjacency Matrices of the Block Diagrams (Real minimum and maximum complexity[21]). This algorithm ensures the successful emergence of the complex product in a multi agent design environment. This is so because the algorithm is in accordance with the recent results that argue for flatter, organic organizational structures that enable workers to deal more effectively with dynamic and uncertain environments (Hinds and McGrath, 2006). It also permits the formation and execution of hierarchical design organizations without the need for the top level managers, coordinators, facilitators, etc.

[21] See Section A.2 for more clarification on minimum and maximum complexity.

7 Product Immunity

So far immunity was studied at the process level and organization level. Immunity of a product that performs in an uncertain environment, as well as the uncertainty introduced to its performance by manufacturing process capabilities, is studied in this Chapter. It was explained in Chapter 1 that complexity and uncertainty together lead to the failure and fragility of a system due to sensitive dependence to small perturbations. This sensitivity dependence or chaos is the reason why complex systems fail. We also posed that bifurcating from one mode of complexity to another mode with higher complexity must be a major concern for complex products. This Chapter presents a global robustness measure as an Immunity Index that can be used in parametric decision making to reduce the likelihood of catastrophic bifurcation to a higher complexity mode (Efatmaneshnik and Reidsema, 2007.a).

7.1 Current Practices of Robust Design

Robustness in general is the insensitivity of some features of a system to uncertainty or, in the case of concurrent design, stochastic perturbations either caused by an uncertain environment in which the artefact is to perform, or by manufacturing and assembly imperfection. Robustness has been discussed, in the past, in parameter design and tolerance design as methods of maintaining the ‘‘reliability’’ (reliability engineering) of a product in the upstream design cycle, whereas on-line process monitoring such as statistical process control methods and failure mode and effect analysis constitute downstream methods of ensuring robustness. Amongst the upstream design cycle methods of robustness, Sensitivity Analysis and Variance Reduction have been the prime tools for robust parameter and tolerance design.

Mellacheruvu et al (2000) used the gradient estimation method to estimate the sensitivity of the manufacturing outputs (such as cost and lead time) to

simultaneous perturbations of some input parameters. He concluded that this method is very effective for, (1) output noise reduction of complex systems, and (2) the identification of the input parameters that have the most impact on the system. Some of the more recent sensitivity analyses, however, have been based on an entropy approach, which can measure the effect of stochastic perturbation on the entropy (a form of global scatter) of the performance parameters.

Fathi and Palko (2001) proposed a signal-to-range ratio for a robust parameter design problem by using Design of Experiments. Liu and Chen (2006) proposed a probabilistic sensitivity analysis for robust design based on the concept of relative entropy to evaluate the impact of a random variable on a performance parameter by measuring the divergence between two probability density functions of the performance response, obtained before and after the variation reduction of the random variable. Robustness indices have been used sporadically to characterize the robustness of different solutions, and mainly as a tool to grade the sensitivity of the performance parameters to other system parameters. For example, Caro et al (2005) proposed a robustness index based on the sensitivity analysis of the performance parameters to the variation of the design parameters. They showed the sum of the Euclidean and Frobenius norms of the sensitivity Jacobian matrix of the design, was an index to quantify the robustness.

These methods, however, are not able to immunize the system against fragility and chaos, or unpredictable failure events. For complex systems in which emergence or collective effect of variables plays a significant role, robust solutions cannot be achieved solely through ‘‘variance reduction’’ which is a reductive approach. Also sensitivity analyses cannot be effective in maintaining robustness at the complex systems level. Sensitivity analysis usually assesses the effect of each individual parameter on each other and ignores their synergies. We need a holistic approach, or at least some combination of reductive and holistic. For this purpose, complexity measures that are based on a holistic representation of systems thinking may be more beneficial.

Process capabilities have been a centre of focus for robust tolerance synthesis (or tolerance design). Since they relate the manufacturing process capabilities to tolerance specifications it is quite natural to apply them to concurrent design of tolerances and processes (Feng and Balusu 2002). Kusiak and Feng (2000) approached robust tolerance design by relaxing the non-critical dimensions to be determined via a combination of Design of Experiments and Monte Carlo Simulation. They also used both worst-case scenarios and response surfaces to calculate tolerance stack up. This kind of tolerance relaxation is not suitable for complex systems, because it may introduce additional complexity modes and global modalities to the system which threatens the immunity of the system.

English and Taylor (1993) used a Kolmogorov-Smirnov distance as a robustness index to characterize the robustness of the process capability ratios. Multimodal variables are those that have two or more most likely values and it is measured by the departure of the variable's probability distribution from the corresponding Gaussian distribution which in statistics is called Kolmogorov-Smirnov distance. They attributed the lack of robustness of manufacturing processes to a large departure of the probability distributions of process capability ratios, C_p and C_{pk}, from normality. In complex systems the most likely value shift or modality of the probability distribution of the system parameters plays a more important role, and can largely affect the collective behaviour of the system. Unlike the variance reduction approach, the modality of variables approach to robustness is a holistic approach because it considers the cause of global bifurcations and emergence of anomalies (Figure 7.1).

Testing the modality of the variables, however, ensures the robustness of the non-self or the environment. The environment of a system or ‘‘non-self ’’ is the set of input and outputs whereas the self of the system is an ‘‘effect’’ on the inputs which results in outputs. In other words the self of the system is the way inputs and outputs interact. The metaphors of ‘‘self ’’ and ‘‘nonself ’’ help us to better understand the dependency of the system-environment and the role of complexity measure in characterizing the ‘‘mode’’ of a system. Testing the robustness of the product's self against catastrophic bifurcation constitutes the

right approach to maintaining the robustness of complex product. We propose to regard the change in the complexity as ‘‘global modality’’ for it refers to the bifurcation of the self as opposed to the modality of the system’s variables.

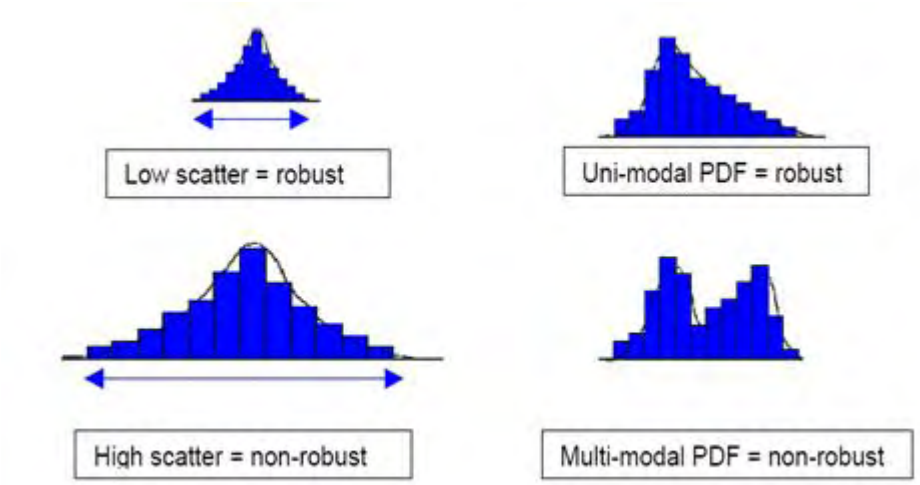


Figure 7.1 Variance Reduction (left) and Modality analysis (right) approaches to robustness.

Marczyk (2002) stated that to make complex systems robust first and foremost consideration should be to avoid optimal solutions. Optimization as performance related decision making technique for satisfying multiple objectives must be forsaken when dealing with complex systems. Marczyk (2002) has stated that:

> Entropy is an ever increasing quantity, and therefore optimum solutions that are in general minimum entropy solutions imply hyper sensitivity to small perturbations. Therefore optimal solutions introduce tremendous amount of fragility into complex systems.

Marczyk (2008) maintained that optimizing an artefact or a system for structural simplicity (least complexity) would make the artefact robust. He also posed that the relative distance between the complexity and the maximum complexity of an artefact as yet another indicator of the robustness. While we

cannot argue against the former, the latter is per se a sort of optimization, and should be avoided. Simply by optimizing for simplicity the system may be temporarily robust but the likelihood of shifting to a high complexity mode (non-robust) is not eliminated. It is more reasonable to avoid design space states in the most complex modes that are potential failure modes.

We expand on the complexity based design method introduced by (Maczyk, 2008) and propose a global Immunity Index for each state of the solution space (FL) (Efatmaneshnik and Reidsema, 2007.a). Marczyk and Deshpande (2006) state that:

> There is a sufficient body of knowledge to sustain the belief that whenever dynamical systems undergo a catastrophe, the event is accompanied by a sudden jump in complexity. This is also intuitive: a catastrophe implies loss of functionality, or organisation.

The Immunity Index can draw comparison between the solutions in terms of their global modality and by that can indicate the immunity of the design solutions (Efatmaneshnik and Reidsema, 2007.a). The solutions should be located in the states that pose the least amount of threat in terms of complexity shift when and if a bifurcation to another complexity mode takes place (Efatmaneshnik and Reidsema, 2007.a). This kind of immunity is a characteristic of the system's self and not its environment and can be most useful in robust parameter and tolerance design of a complex system such as concurrent product-process design.

7.2 A Global Robustness Index Based on Complexity Modes

Considering a d dimensional FL with N fuzzy levels. This FL would have N^d fuzzy states. We define a complexity gradient for the nth state of the FL as (Efatmaneshnik and Reidsema, 2007.a):

$$\nabla C_n = \underbrace{\left(\frac{\partial C_n}{\partial X_1} + \frac{\partial C_n}{\partial X_2} + \cdots + \frac{\partial C_n}{\partial X_d}\right)}_{a} + \underbrace{\left(\frac{\partial^2 C_n}{\partial X_1 \partial X_2} + \frac{\partial^2 C_n}{\partial X_1 \partial X_3} + \cdots + \frac{\partial^2 C_n}{\partial X_{d-1} \partial X_d}\right)}_{b} + \cdots + \underbrace{\frac{\partial^d C_n}{\partial X_1 \cdots \partial X_d}}_{c} \quad (1)$$

Where $X_i s$ are system parameters including inputs and outputs. The term *(a)* is the sum of all changes in complexity due to the perturbation of only one variable while all other variables are unchanging (have no perturbation). The term *(a)* estimates the amount of change in complexity for single parameter perturbation. The term *(b)* estimates the amount of change in complexity for the simultaneous perturbations of any two arbitrary set of variables. The term (c) calculates the same sum for simultaneous perturbations of all *d* variables at the same time. For simplification let's assume that a random perturbation would be no more than one fuzzy level and then all $\partial X_i s$ would be equal to *1*. Thus we have (Efatmaneshnik and Reidsema, 2007.a):

One fuzzy level to the left — One fuzzy level to the right

$$\frac{\partial C_n}{\partial X_i} = \frac{\overbrace{\frac{\partial C_n}{\partial X^-_i}} + \overbrace{\frac{\partial C_n}{\partial X^+_i}}}{2^1 - I} \quad (2)$$

Where:

$$\frac{\partial C_n}{\partial X^-_i} = C_{n'} - C_n \quad \text{and} \quad \frac{\partial C_n}{\partial X^+_i} = C_{n''} - C_n \quad (3)$$

n' and n'' are respectively the left and right states of the state *n* along the X_i axis. *I* is the number of impossible states amongst the 2^p states which are the states that contain no data points. The derivatives related to these states must be eliminated from these equations. If *I* equals to *2* then no perturbation could occur for variable X_i. Figure 7.2 clarifies this: each arrow in left anthill plot of Figure 7.2 represents the state transition as the result of single variable perturbation (left-and-right or up-and-down).

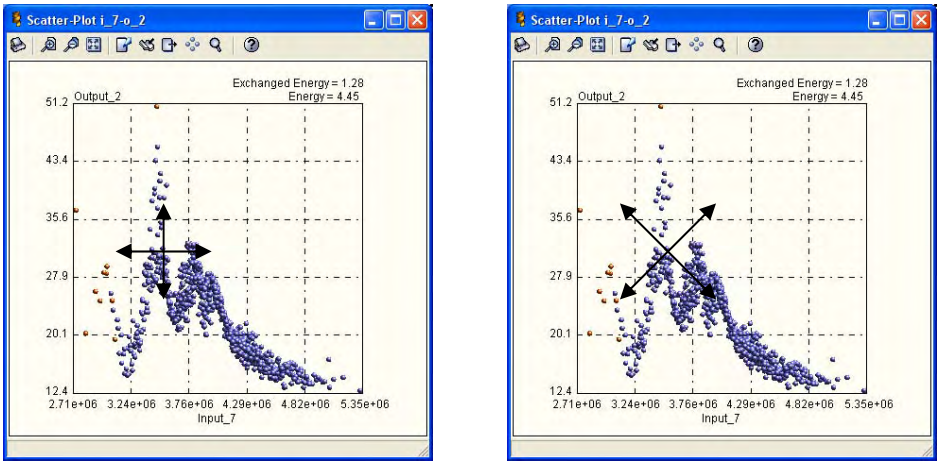


Figure 7.2 Perturbation of an initial state to neighboring states.

Equations (4) and (5) formulate the sum of all likely changes in complexity by simultaneous disturbance of two variables (Efatmaneshnik and Reidsema, 2007.a):

$$\frac{\partial^2 C_n}{\partial X_1 \partial X_2} = \frac{\dfrac{\partial^2 C_n}{\partial X_1^- \partial X_2^-} + \dfrac{\partial^2 C_n}{\partial X_1^+ \partial X_2^-} + \dfrac{\partial^2 C_n}{\partial X_1^- \partial X_2^+} + \dfrac{\partial^2 C_n}{\partial X_1^+ \partial X_2^+}}{2^2 - I} \tag{4}$$

$$\frac{\partial^2 C_n}{\partial X_1^- \partial X_2^-} = C_{n'''} - C_n \tag{5}$$

$$\vdots$$

where n''' is the state down and left to the state n, and so on.

In the right scatter plot of Figure 7.2 arrows show the possible state transitions as result of simultaneous perturbation of two variables. The remaining derivatives are calculated in the same way. In general, for simultaneous perturbation of p parameters at the same time we have (Efatmaneshnik and Reidsema, 2007.a):

$$\frac{\partial^{p}C_{n}}{\partial X_{1}\partial X_{2}\cdots\partial X_{p}}=\frac{\frac{\partial^{p}C_{n}}{\partial X_{1}^{-}\partial X_{2}^{-}\cdots\partial X_{p}^{-}}+\frac{\partial^{p}C_{n}}{\partial X_{1}^{+}\partial X_{2}^{-}\cdots\partial X_{p}^{-}}+\frac{\partial^{p}C_{n}}{\partial X_{1}^{+}\partial X_{2}^{+}\cdots\partial X_{p}^{-}}+\cdots\frac{\partial^{p}C_{n}}{\partial X_{1}^{+}\partial X_{2}^{+}\cdots\partial X_{p}^{+}}}{2^{p}-I} \quad (6)$$

$$\frac{\partial^{p}C_{n}}{\partial X_{1}^{-}\partial X_{2}^{-}\cdots\partial X_{p}^{-}}=C_{m}-C_{n} \quad (7)$$

$$\vdots$$

Where C_m is the complexity of state m at which p parameters of the system have shifted one fuzzy level relative to the initial state (n). Finally the Immunity index for the state n of the FL is defined as (Efatmaneshnik and Reidsema, 2007.a):

$$R_{n} = (1- (\frac{\nabla C}{(m-1)\times C_{n}}))\times 100\% \quad (8)$$

Where m is calculated as:

$$m = \binom{N}{N-1} + \binom{N}{N-2} + \cdots + \binom{N}{0} - \{I_1 + I_2 + \cdots + I_N\} \quad (9)$$

I_i s are the number of impossible states for the perturbation of i parameter at the same time. An Immunity Index close to *100%* represents the situation that the least amount of likelihood exists for the complexity of the state to change with any stochastic perturbation of the system parameters, which implies that the state of the system is globally uni-modal, robust and immune from increasing in complexity (Efatmaneshnik and Reidsema, 2007.a). An Immunity Index value close to zero shows that there is a large chance of complexity increase or catastrophe by stochastic perturbation, i.e.; the state of the system is globally multi-modal and non-robust (Efatmaneshnik and Reidsema, 2007.a). A value greater than *100%* implies that a bifurcation is even less likely to produce a catastrophe and that there are chances of a decrease in complexity (Efatmaneshnik and Reidsema, 2007.a).

7.3 Discussion: Complexity Based Decision Making

This approach, in fact, is the sensitivity analysis of the complexity mode of a system to random perturbations of the system parameters. The Immunity Index gives the estimation of the robustness of every state of the FL based on its tendency to change its complexity mode with random perturbations of the system parameters. Entering to a new complexity mode implies bifurcation whether it increases the complexity or not. The complexity increase as a result of entering to a new state should be taken as a catastrophe. A state in the FL is globally robust if all of its neighbouring states are located in the modes with the same or less complexity (Efatmaneshnik and Reidsema, 2007.a). Thus, the state closer to the centre of the mode has more robustness and is immune from jumping to other modes (Efatmaneshnik and Reidsema, 2007.a).

The global Immunity Index in effect indicates the above fact by testing the complexity modes of all neighbouring states of a given fuzzy state of the FL (Efatmaneshnik and Reidsema, 2007.a). For multi-modal solutions a stochastic perturbation in the system's state increases the complexity of the system, thus, chaotic behaviour becomes more likely (Efatmaneshnik and Reidsema, 2007.a). We say, in this case, that the system is globally sensitive and unstable with regard to stochastic perturbations (Efatmaneshnik and Reidsema, 2007.a).

In summary, the Immunity Index is a form of objective or cost function that requires the absence of any other objective function (Efatmaneshnik and Reidsema, 2007.a). Rather the performance objectives should be regarded as performance constraints (Efatmaneshnik and Reidsema, 2007.a). After determining the design variables and their constraints, the FL containing the set of solutions corresponding to those constraints must be simulated (Efatmaneshnik and Reidsema, 2007.a). Then the decision making must be based on the rule of lower complexity, and the first step is to choose a mode in the landscape that has a low enough complexity (Efatmaneshnik and Reidsema, 2007.a). Each mode usually contains many states of the system, each of which can be regarded as the solution (Efatmaneshnik and Reidsema, 2007.a). The solution should be located in a state that has the maximum immunity and that is exactly at the core or centre of

the multi dimensional mode; this leads to little chance that random perturbations will move the state of the system to another mode with higher complexity (Efatmaneshnik and Reidsema, 2007.a). The global robustness index must be calculated for every fuzzy state of the complexity mode (Efatmaneshnik and Reidsema, 2007.a).

8 A Decision Support System: IMMUNE

"There is a substantial amount of empirical evidence that human intuitive judgment and decision making can be far from optimal, and it deteriorates even further with the complexity of the problem and stress" (Druzdzel and Flynn, 2000). Decision making under these conditions requires support in the estimation, the evaluation and/or the comparison of alternatives (Turban, 1995). A DSS is a "knowledge-based system, which formalizes the domain knowledge so that it is amenable to mechanized reasoning" (Druzdzel and Flynn, 2000). More specifically a DSS is an integrated, interactive, flexible, and adaptable computer-based computing environment, especially developed for supporting the solution of a non-structured complex management problem (Turban, 1995; Druzdzel and Flynn, 2000).

Knowledge-based problem solving is the domain of Artificial Intelligence (AI) and the selection of an appropriate AI development tool that may provide a framework to incorporate knowledge will come from this area (Reidsema and Szczerbicki, 2002). Reidsema and Szczerbicki (2002) identified three different architectures for DSS for product design planning and manufacturing in a concurrent engineering environment: Expert Systems, Agent Based Systems, and Blackboard Database Systems. These have been defined as follows:

- An *Expert system* is one of a class of AI techniques that is able to capture the knowledge and reasoning of an experienced expert for re-use in assisting the less experienced in making decisions.
- The *Blackboard Database Architecture* (BBDA) is a problem solving system based on the metaphor of human experts who cooperate by entering partial solutions to the current problem onto a physical blackboard. The type of problems best suited to this approach is those that are able to be reduced to a set of simpler problems that are reasonably independent. The integration of the partial solutions to the overall solution

takes place by the intervention of a centralized controller known as a control source and therefore has a top down approach to problem solving.

- *Multi agent systems* are distributed systems that use a bottom up approach to problem solving in which case the intervention of the centralized coordination between agents is minimal or totally eliminated. Each agent in a multi agent system behaves as an abstraction tool which has the characteristics of a self-contained problem solving system that is capable of autonomous, reactive, proactive as well as interactive behaviour. The solution in this case emerges as a whole and is the result of a synergetic effect. “Synergy denotes a level of group performance that is above and beyond what could be achieved by the members of the group working independently” (Larson, 2007). Synergy in a multi agent system enables the integration of partial solutions of nonlinear and coupled problem. Multi agent systems are the natural candidate for complex systems which show heavy interdependency between partial problems.

Providing an extensive literature review of concurrent design and manufacturing systems, Shen et al (2001) identifies three different approaches for agent based architectures: hierarchical architectures, federated architectures and autonomous agent architectures. Each architecture has particular strengths for specific applications and choosing the right architecture involves matching requirements to capabilities. Hierarchical architectures consist of semi-autonomous agents with a global control agent dictating goals/plans or actions to the other agents. Multi-Agent Systems with a global blackboard database are hierarchical architectures. According to Shen et al (2001) some researchers have considered their blackboard systems to be multi-agent systems, and others have implemented their agent based systems using blackboard architectures. In these systems, control can be implemented in different ways: using a special control expert called a supervisor as in EXPORT (Monceyron and Barthes, 1992): using a shared graphical model as in ICM (Interdisciplinary Communication Medium) (Fruchter et al, 1996) or a shared database as in SHARED (Wong and Sriram, 1993); or through multiple shared workspaces as in MATE (Saad and Maher, 1996).

Because hierarchical architectures suffer from deficiencies associated with their centralized character, federated multi agent architectures are increasingly being considered as a compromise solution for industrial agent based applications, especially for large scale engineering applications (Shen et al, 2001). A fully federated agent based system has no explicit shared facility for storing active data; rather, the system stores all data in local databases and handles updates and changes through message passing (Shen et al, 2001). In theory, a truly open multi agent system need not have any predefined global control (Shen et al, 2001). An example of such architecture is DIDE (Distributed Intelligent Design Environment) (Shen and Barthes, 1996). Another good example of such a system is ANARCHY which was a working prototype of an asynchronous design environment (Quadrel et al, 1993). Agents in ANARCHY were autonomous, and used broadcast communications. It however, utilised a global design strategy based on simulated annealing. For such systems there is the threat of exhibiting chaotic behaviour (Sycara, 1998).

This Chapter presents IMMUNE which is a flat federated architecture for the parametric design of complex products (Efatmaneshnik and Reidsema, 2009). IMMUNE uses a global blackboard to save the current state of the design (Efatmaneshnik and Reidsema, 2009). All the agents are grouped into virtual teams or coalitions in a way that the design system becomes capable of mirroring the structure of the problem and its decomposition pattern at each abstraction level (Efatmaneshnik and Reidsema, 2009). This idea was previously, to some extent, utilized in MetaMorph (Maturana et al, 1999) that was devised “as an adaptive agent based architecture to address system adaptation and extended-enterprise issues at four fundamental levels: virtual enterprise, distributed intelligent systems, concurrent engineering, and agent architecture”. MetaMorph was a federated architecture that could dynamically adapt to the tasks and changing environment by using dynamically formed agent groups (Shen et al, 2001). The agents with various knowledge and utility were clustered into virtual groups or coalitions. In MetaMorph, resource agents could be cloned as needed for concurrent information processing (Maturana et al, 1999). The clone agents were

included in the virtual clusters. MetaMorph benefited extensively of low level mediators to coordinate between different groups (or coalitions).

IMMUNE, however, does not have any low level mediator, broker or facilitator, and is a flat architecture (Efatmaneshnik and Reidsema, 2009). Instead IMMUNE benefits from a special unit in the control shell of the blackboard to which we denote as the CEO (Complexity Estimator and Observer) (Efatmaneshnik and Reidsema, 2009). The CEO estimates the complexity measure of the problem and compares it to the observed complexity of the multi agent system that is raised from agents' interactions (Efatmaneshnik and Reidsema, 2009). Based on the complexity of the problem after decomposition (real complexity), the CEO estimates the minimum and maximum complexity of the process (Efatmaneshnik and Reidsema, 2009). It then monitors the complexity of the process as a function of the exchanged information between the agents (cognitive complexity) (Efatmaneshnik and Reidsema, 2009). An *effective* and *efficient* design process must have a cognitive complexity in between the minimum and maximum complexity of the problem (Efatmaneshnik and Reidsema, 2009); this is the result of a simple notion which is: the best a single person (or a single system) can do is limited by his/her (cognitive) complexity (Bar-Yam, 2004).

A design system, with its cognitive complexity surpassing the maximum complexity of the problem, has lost *effectiveness* since the design process may become chaotic (Efatmaneshnik and Reidsema, 2009). If the cognitive complexity of the design system is lower than the minimum complexity of the problem, then the *efficiency* of the system, in solving the complex problem and managing the interdependencies between its sub-problems, would not be achieved In both cases, the agents are expected to undertake corrective measures to stabilize the cognitive complexity of the system and immunize it against fragility, and failure (Efatmaneshnik and Reidsema, 2009). The CEO monitors the complexity at two levels: inside the coalitions (at the local levels) and the entire system (at the federal level) (Efatmaneshnik and Reidsema, 2009). The next section discusses the fundamentals of design planning for complex products and a complexity based method for monitoring the design process.

8.1 Collaborating Architecture

Shen et al (2001) states that:

> Real world concurrent engineering design projects require the cooperation of multidisciplinary design teams; individuals and multidisciplinary design teams work in parallel with various engineering tools that are located at different sites often for quite a long time. To coordinate the design activities of various groups and guarantee good cooperation among the different engineering tools a distributed intelligent environment is required.

Such an environment requires the utilization of *collaborating software* as a DSS that involves the integration and coordination of relatively independent, self-contained software systems that are able to work together effectively on their own (Corkill, 2003). "Collaborating software is very different from *collaboration software*, where the software is used to facilitate the interaction among human participants rather than to provide an automated environment where software—and potentially human—entities work together in order to perform complex activities" (Corkill, 2003). For effective development of collaborating software Corkill (2003) identified six main challenges:

> 1. *Representation:* To enable system components and modules to understand one another.
> 2. *Awareness:* to render modules aware when something relevant to them occurs.
> 3. *Investigation:* helping modules to quickly find information related to their current activities.
> 4. *Interaction:* to create modules that are able to use the concurrent work of others while working on a shared task.
> 5. *Integration:* to combine results produced by other modules.
> 6. *Coordination:* ensuring that modules focus their activities on the right things at the right time.

Except the representation that is the main challenge is collaboration software, remainders of these challenges are addressed in IMMUNE. So far two main approaches have been considered for the design of a collaborating environment: Blackboard architectures and Multi Agent architectures. Corkill (1991) presented the following metaphor to describe the Blackboard-based problem solving:

> imagine a group of human specialists seated next to a large blackboard. The specialists are working cooperatively to solve a problem, using the blackboard as the workplace for developing the solution. Problem solving begins when the problem and initial data are written onto the blackboard. The specialists (knowledge sources) watch the blackboard, looking for an opportunity to apply their expertise to the developing solution. When a specialist finds sufficient information to make a contribution, she records the contribution on the blackboard, hopefully enabling other specialists to apply their expertise. This process of adding contributions to the blackboard continues until the problem has been solved

Each problem solving expert is designed to independently contribute specialized knowledge required to solve one aspect of the overall problem. The sequence of the contributions of these experts is not determined *a priori* but is instead based on the current state of the solution and the selection of the most applicable and effective expert for solving the associated problem part (Reidsema, 2001). As such, the blackboard model of problem solving is a highly structured case of opportunistic problem solving (Reidsema, 2001).

The Blackboard architectures utilize abstraction and solve problems through iteration. Blackboard architectures are able to maintain the focus of attention of different knowledge sources asynchronously on different abstraction levels within this memory; As a result the design knowledge of various parts of a

product in various abstraction levels can be considered together. This means that the hitch that Bar-Yam (2004) (mentioned in Chapter 2) has maintained about the abstraction for design of complex systems is not relevant when Blackboard databases are used The downside with purely blackboard architectures however is that the knowledge sources (design players) do not communicate with each other directly and communication is solely done through the blackboard. As such blackboard systems suit only the loosely coupled problems (Corkill, 2003). On the other hand multi agent systems due to their ability to interact autonomously can reach to high overall cognitive complexity to solve densely interconnected problems without the need for a global integrator.

Multi agent systems on the other hand have the following problem solving characteristics (Corkill, 2003):

- Distribution (no central data repository)
- Autonomy (local control)
- Interaction (communication and representation)
- Coordination (achieving coherence in local control decisions)
- Organization (emergent organizational behaviour)

Multi Agent System architectures are expressed as the pattern of relationships amongst agents (Shen et al, 2001). Two kinds of relationships may be supposed between agents: Control relationships and Collaboration relationships (Shen et al, 2001). A Control relationship relates to the degree of autonomy which an agent possesses. An agent whose goals, plans and/or actions are prescribed by the imperatives of another agent(s) has little autonomy. In a collaborating relationship however, the agents involved are free to accept, reject or modify goals, plans or actions proposed to them. In theory, a truly open multi agent system need not have any predefined global control. An example of such architecture is that of DIDE (Shen and Barthes, 1996).

According to Corkill (2003) a quarter-century of blackboard-system experience and more than a decade of multi agent system development have produced a strong baseline of collaborating-software technologies. The next

generation of complex, collaborating software applications must span the entire design space of Figure 8.1 to enable development of high performance, generic collaborating-software capabilities. This is the motivation behind the design of the presented architecture in this paper (IMMUNE) which combines the agent based and blackboard technologies and remains very uncommon to date. Collaboration in the combination of multi agent blackboard environments can be asynchronous (through the blackboard) and not restricted to one abstraction level, as well as autonomous direct communication. This rare approach first started by Lander et al (1996) proposing to use agent based blackboards to manage agent interactions (Shen et al, 2001). Their model contained multiple blackboards, used as data repositories for each group of agents. Along with design data, tactical control knowledge could be represented in the shared repository, enabling reasoning about the design itself (Shen et al, 2001). SINE (Brown et al, 1995) was another agent based blackboard platform that used a single global blackboard to record the current state of the design. Even though agents could exchange messages directly, design data could flow through the blackboard, and it was accessible to all agents (Shen et al, 2001).

Our proposed architecture (IMMUNE) is an agent based blackboard system that uses a flat and federated architecture (Efatmaneshnik and Reidsema, 2009). All the agents are grouped into virtual teams or coalitions. There is no local controller for coordination in between the coalitions. IMMUNE uses a global blackboard to save the current state of the design and to facilitate asynchronous communication between agents through the blackboard by saving the complete solution and partial solutions of different abstraction hierarchies.

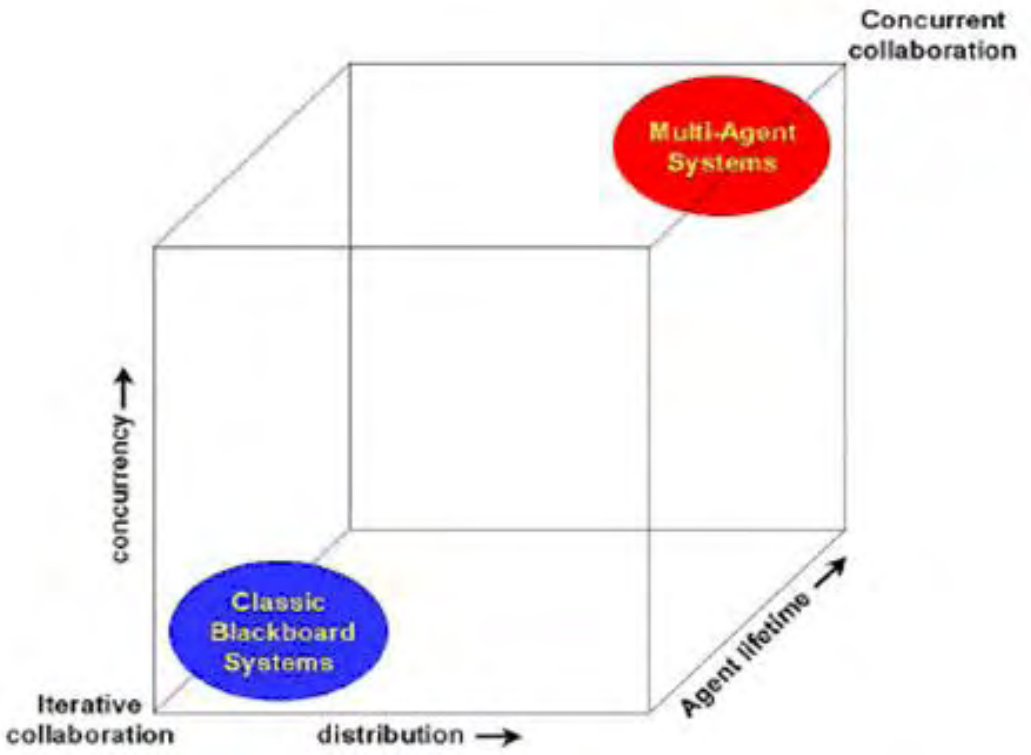


Figure 8.1 Collaborating environment comparison, after Corkill (2003).

The primary purpose of designing this architecture was to incorporate the complexity science into the collaborating software paradigm. Lissack (1999) demonstrated that since both organization science and complexity science deal with uncertainty it is important to combine the two. This marriage of the two sciences allows for having an autopoietic view to organization. Autopoietic systems theory, analyse systems as having self-productive, self-organized, and self-maintained nature (Dissanayake and Takahashi, 2006). The main characteristic of IMMUNE is the digestion of complexity measures of the product and process which enables the manifestation of autopoietic characteristics (Efatmaneshnik and Reidsema, 2009).

The control source of the proposed blackboard does not dictate the pattern of cooperation between agents allowing autonomy in the interaction. It however does monitor the complexity of the system at two levels: inside the coalitions and in between the coalitions at the same abstraction level (we refer to this as a layer). The agents are designed to react to the information that they receive from the control source about complexity of the coalitions and layers (Efatmaneshnik and

Reidsema, 2009). Adding or eliminating agent(s) from the design system is possible in IMMUNE, making it an open architecture.

8.2 Blackboard Architecture

The Blackboard Database is a hierarchical and partitioned global memory space that acts as a central storage area for holding problem solving data, information and partial solutions that represent the problem to be solved (Craig, 1993). The blackboard provides a common data structure that acts as an interface to agents (or knowledge sources in standard blackboard systems) allowing them to read the problem data and alter the state of this data when necessary, thereby effecting an incrementally improved solution to the problem (Reidsema, 2001). Abstraction levels are introduced on the blackboard gradually. Each abstraction hierarchy is decomposed according to one of the modes described earlier in Chapter 4. The envisaged IMMUNE's blackboard contains the design variables, their interactions (derived from simulation) and the agents' proposed solutions (Efatmaneshnik and Reidsema, 2009).

In IMMUNE the Blackboard facilitates the measurement of problem complexity at three levels: sub-problems in one abstraction level, problems of one abstraction level and global problem complexity between all abstraction levels (Efatmaneshnik and Reidsema, 2009). All of these three complexities change with time (progression of the design process). As the agents generate a design variable, the problem complexity and global problem complexity can only increase. In case the value of a design variable is resolved the problem complexity and the global problem complexity can only decrease. The global complexity, in particular, provides the opportunity to monitor temporally distributed problem solving that takes place in the form of asynchronous information exchange between various abstraction levels.

The blackboard also facilitates incorporation of the Immunity Index presented in Chapter 7 in the decision making process. The complexity modes can be estimated at the three levels of subsystems in one abstraction level, a given

abstraction level and the entire system. The decision makers can immediately estimate the effect of their choices on the immunity/robustness of the whole product/artifact.

8.3 Control Source

Typically blackboard architectures provide a control mechanism called a control source to coordinate the use of knowledge sources in a consistent and effective manner (Reidsema, 2001). The control source determines which knowledge sources should make a contribution to the solution, when it should do so, and what part of the solution should be the focus (Efatmaneshnik and Reidsema, 2009). In IMMUNE, however, the agents decide their focus of attention in a manner described in the next section (Efatmaneshnik and Reidsema, 2009). The control source of IMMUNE comprises several agents with distinct tasks, all of which can be computationally modeled (Efatmaneshnik and Reidsema, 2009).

8.3.1 Decomposition Agent

Decomposition agent decomposes the generated problem on the main blackboard according to the connectivity of the problem (as was discussed in Chapter 4). Important control features that affect the entire system's performance can be incorporated in this agent's knowledge namely the number of subsystems, and decomposition mode (Efatmaneshnik and Reidsema, 2009).

8.3.2 Composition Agent

Composition Agent groups the agents based on their bids for the problems in coalitions using the contract net protocol (Efatmaneshnik and Reidsema, 2009). Composition agent contains the map of all the agents, their characteristics and types of expertise (Efatmaneshnik and Reidsema, 2009).

8.3.3 IT manager

IT manager sets up the LAN and communications channels of the dispersed agents for each abstraction level (Efatmaneshnik and Reidsema, 2009). All the agents in the same coalition must be visible to each other meaning that the messages that one agent receives is made visible to all team members (Efatmaneshnik and Reidsema, 2009). This may be thought of as a shared mailbox for each coalition (Efatmaneshnik and Reidsema, 2009). For example, Figure 8.2 shows that when a message is sent from agent 2 in coalition A to agent 4 in coalition B, it would be visible to all the members of these two coalitions.

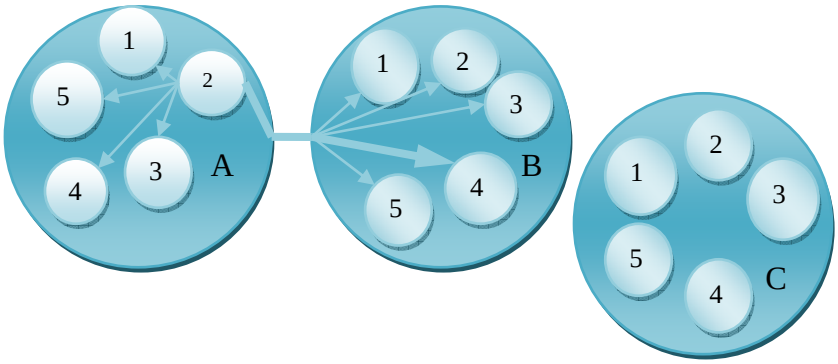


Figure 8.2 Shared mail boxes for coalitions.

8.3.4 Simulation and Computation Agent

Simulation Agent performs Monte Carlo Simulation to generate the design space FL, and tags each state of the FL with its immunity index. The agent comprises a Monte Carlo Simulation software package, Ontospace™ software. It gathers information about the conditional probability distribution of the design variables from the agents that generate them. This agent, runs the Ontospace™ software using the generated FL, extracts the complexity modes and gives a tag to each state of the FL. The agent performs all the above at three levels of the subsystems in one abstraction level, the system at one abstraction level and the entire system of systems at all abstraction levels that are introduced up to then. This agent must be able to dynamically simulate the FL when the new entries (design variables) appear on the blackboard for a given abstraction level and also when design

values are finalized by the design agents. Since there is no prerequisite for the activation of one abstraction level, this agent must be able to run parallel simulations for two or more abstraction levels in one time (Efatmaneshnik and Reidsema, 2009).

8.3.5 CEO (Complexity Evaluator and Observer)

This agent announces the termination of the generation stage for a given abstraction level as soon as the complexity of the level reaches a certain threshold (Efatmaneshnik and Reidsema, 2009). This threshold is a control feature of the entire system. As mentioned before the termination of the generation stage in each abstraction level, can be conditional to the self size. This provides control over the size of the problem space (or FL). The knowledge of the ideal abstraction approach of Liu et al (2003) can be incorporated into the CEO by constraining the number of design variables at each abstraction level.

The CEO agent also monitors the design process (Efatmaneshnik and Reidsema, 2009). It has an embedded blackboard on which all the communications between the agents and in between the agents and blackboard are recorded (Figure 8.3). The design agents can only write on this blackboard but there is no necessity for them to be able to read it (Efatmaneshnik and Reidsema, 2009). The communication arrows on this blackboard must have a tag that represents both the qualitative and quantitative weight of transferred information (Efatmaneshnik and Reidsema, 2009). Based on these maps of the system which vary regularly over time, the CEO measures the instantaneous cognitive process complexity of each coalition (a team of agents), a layer (in between the coalitions of one abstraction level), and the global cognitive complexity (in between the layers that are temporally distributed) (Efatmaneshnik and Reidsema, 2009). The CEO measures the lower and upper complexity bounds at all these three levels (Efatmaneshnik and Reidsema, 2009). The control source must contain knowledge of the different types of resource agents in terms of their capabilities, functionalities and discipline knowledge. If an agent is being recruited to the

system it must register all its characteristics with the composition agent of the control source (Efatmaneshnik and Reidsema, 2009).

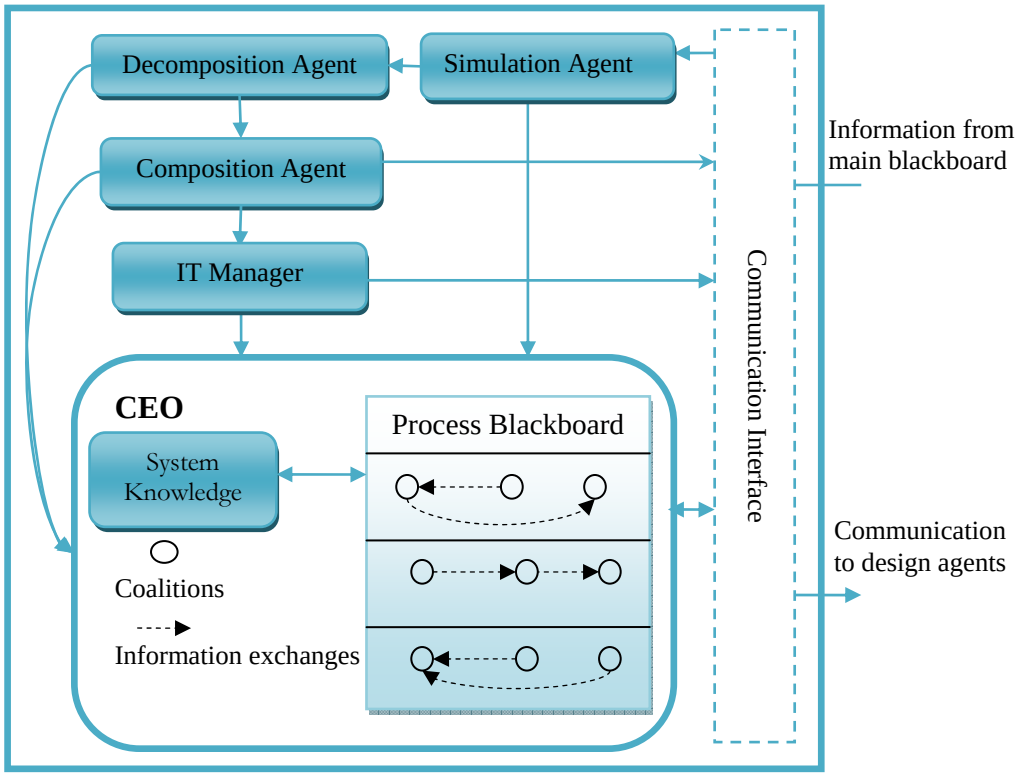


Figure 8.3 The control source structure of IMMUNE.

The control source may be fully computational and may not need any human intervention to proceed with its tasks (Efatmaneshnik and Reidsema, 2009). The design process begins with the control source broadcasting notices to all agents with regard to the generation of new design variables (Efatmaneshnik and Reidsema, 2009). The agents place their entries on the blackboard in the specified abstraction level. This is the generation stage of the presented design template.

The first set of initial design variables act like a seed on the blackboard that would gradually evolve to other design parameters at other abstraction levels (Efatmaneshnik and Reidsema, 2009). The control source, then, whether by itself or through knowledge sources, is in charge of simulating the FL (or design space) corresponding to these sets of design parameters and the extraction of the self map

for the design parameters (Efatmaneshnik and Reidsema, 2009). After this, the control source decides on the number of sub problems, and the decomposition mode, decomposes the self by considering the number of active design agents (design resources) and the real complexity (Efatmaneshnik and Reidsema, 2009). The control source then clusters the design agents into virtual teams and distributes the sub-problems to them (Efatmaneshnik and Reidsema, 2009). The design agents within the virtual teams solve the problems cooperatively. They send the results back to the blackboard, and negotiate the conflicts with the other groups until they reach a resolution.

Using the common practices of sequential engineering would lead to starting a new cycle (at new abstraction level) after all activities of the previous cycle are performed and the results are finalized. However, the concurrent engineering principle of overlapping the activities to shorten the design lead time may be applied here. Therefore agents must be allowed to introduce their proposed design variables on the blackboard (problem generation), however the decomposition and distribution stages start only when the CEO supposes an abstraction level as having reached a certain complexity threshold. Since agents can be cloned to perform different tasks, it is possible that two or more abstraction levels could simultaneously be at different stages of design process. In order to reduce the complexity of the entire design system, we propose that the design agents of different abstraction levels be allowed to communicate only through the blackboard.

8.4 Agents Structures

In artificial intelligence, an intelligent agent observes and acts upon an environment, as a rational agent: an entity that is capable of perception, action and goal directed behavior. Such an agent might be a human, computer code or an embedded real time software system. The internal architecture of an agent is essentially the description of its modules and how they work together (Shen et al, 2001): agent architectures in various agent based systems (including agent based concurrent design and manufacturing systems) range from the very simple (a

single function control unit with a single input and output) to very complex human like models. The agents in IMMUNE are Single Function Agents (SiFAs) which were developed in the AIDG research group (Dunskus, 1994) to investigate concurrent engineering design problem solving using multi agent architectures. It involves multiple agents that cooperatively produce a solution when the task of the entire system is decomposed into many, very small subtasks; where exactly each one of these is assigned to an individual agent (Dunskus, 1994). Every agent now has one function to perform, that is, to execute its subtask. Agents have their own point of view that represented the expertise of the agent; which might be cost, strength, manufacturability etc (Dunskus, 1994). In IMMUNE, SiFAs have three functions: 1) to generate the design variable; 2) to estimate the values of the design variables; and 3) to evaluate the solutions of other agents from their own point of view, verifying the existence of any conflicts (Efatmaneshnik and Reidsema, 2009). SiFAs are collaborative agents (also called interacting or social agents) that work together to solve problems. The joint expertise of collaborative agents is applied to ensure that the overall design is consistent (Shen et al, 2001).

"Coherence is a global (and regional) property of the MAS that could be measured by the efficiency, quality, and consistency of a global solution (system behavior) as well as the ability of the system to degrade gracefully in the presence of local failures" (Sycara, 1998). Coherency is about the ability of the MAS's to "cope" with problem integration. Several methods for increasing coherence have been studied, all of which relate to the individual agent's ability to reason about the following questions: who should I interact with? And when should I do it and why? Sophisticated individual agent reasoning can increase MAS coherence because each individual agent can reason about non-local effects of local actions, form expectations of the behavior of others, or explain and possibly repair conflicts and harmful interactions (Sycara, 1998). On this basis, four different agent architectures have been discussed in the literature: reactive agents (also known as behavior based or situated agent architectures), deliberative agents (also called cognitive agents, intentional agents, or goal directed agents), collaborative agents (also called social agents or interacting agents), and hybrid agents (Shen et al, 2001).

Reactive agents are passive in their interactions with other agents. They do not have an internal model of the world and respond solely to external stimuli. They respond to the present state of the environment in which they are situated and do not take history into account or plan for the future (Sycara, 1998). Through simple interactions with other agents, complex global behaviour can emerge. In reactive systems, the relationship between individual behaviours, environment, and overall behaviour is not understandable (Shen et al, 2001). However, the advantage of reactive agent architecture is simplicity (Shen et al, 2001).

Deliberative agents use internal symbolic knowledge of the real world and environment to infer actions in the real world. They proactively interact with other agents based on their sets of Beliefs, Desires and Intentions (BDI system). These agents perform sophisticated reasoning to understand the global effects of their local actions (Sycara, 1998). Consequently, they have difficulties when applied to large complex systems due to the potentially large symbolic knowledge representations required (Sycara, 1998). Shen et al (2001) identified collaborative agents as a distinct class of agents that work together to solve problems; communication in between them leads to synergetic cooperation, and emergent solutions.

Hybrid architectures are neither purely deliberative nor purely reactive (Sycara, 1998), and the agents in IMMUNE have a hybrid architecture (Figure 8.4). According to Sycara (1998) hybrid agents usually have three layers:

1. The Lowest Layer: at the lowest level in the hierarchy, there is typically a reactive layer, which makes decisions about what to do on the basis of raw sensor input. This layer contains the self knowledge that is the knowledge of the agent about itself including physical state, location, and skills, etc (Shen et al, 2001).
2. The middle layer: this layer typically abstracts away from raw sensor input and deals with a knowledge-level view of the agent's environment, often making use of symbolic representations (Sycara, 1998). This layer contains two types of knowledge: domain knowledge and common sense knowledge. The domain knowledge is the

description of the working projects (problems to be solved), partial states of the current problem, hypothesis developed and the intermediate results (Shen et al, 2001).

3. The uppermost layer: this layer handles the social aspects of the environment (Sycara, 1998).

The detail of these layers with their modules in design agents of IMMUNE is presented next and depicted in Figure 8.4.

8.4.1 The Lowest Layer

This layer contains the agent's self knowledge that is used to participate in tasks and reply to other agents (through the uppermost social layer) as well as control source's requests about its competence, and capabilities in the bidding process. The main responsibility of the low layer, in fact, is to decide on the values of the design variables. In doing so it checks the immunity of each state of the FL (note that states are already tagged with their immunity). At this layer the agent decides on the potential solutions for the design variables and sends them to the two upper layers to study the potential conflicts that these solutions may have with the objectives of other design agents and for further negotiations with them.

8.4.2 The Middle Layer

The middle layer has two modules, namely agenda manager and COPE (Complexity Oriented Problem Evaluator) that respectively contain the above two types of knowledge and are in contact with backboard and the CEO.

Agenda Manager

Agenda manager is in direct contact with the lower layer. It decides on the focus of attention and reports it to the lower layer. The agenda manager in the middle layer regularly monitors the blackboard to maintain its domain knowledge. Agenda manager also receives the potential solutions from the lower layer and is alert on the reports of the conflicts that the potential solutions have had with the

solutions of other design agents (these reports come into the middle layer directly from the communication interface of the agent. Agenda manager has been used in many blackboard systems such as HEARSAYII (Carver and Lesser, 1994) with the difference that in these systems a central agenda manager has been in charge of maintaining the focus of attention for the entire set of knowledge sources (agents). Generally agenda managers are data driven (what is present on the blackboard) and its operation leads to opportunistic problem solving (Carver and Lesser, 1994). The agenda manager chooses the focus of attention of the agent on different problems at different abstraction levels. It may shift the focus of attention of the agent from one abstraction level to another depending on the status of the problems and partial solution on the blackboard. The main reason for using agendas for control is to speed up the process of problem solving, and for reducing agent idle time (Carver and Lesser, 1994).

COPE

The common sense knowledge that enables the agent to make sense of the cognitive complexity measure of the environment that is reported by the CEO is embedded in this module. The COPE module maintains the cognitive complexity of the one coalition, group of coalitions in one abstraction level and the entire system of active coalitions in all abstraction levels in the appropriate range (Efatmaneshnik and Reidsema, 2009). COPE can make sense of the environment's cognitive complexity (at all three levels) by comparing it to the maximum and minimum complexities that is determined by CEO. COPE is a goal driven module and communicates with the agent's upper layer. To increase the complexity of the environment COPE informs the upper layer of the agent to socialize and collaborate more actively. To decrease the complexity of the environment the upper layer of the agent must become more passive by simply reacting to the incoming information from the control source and other agents. In this way COPE may provide immunity from agent overreacting or under acting in the environment. Also COPE can dictate the upper layer to choose different conflict resolution schemes that are more passive like constraint relaxation to reduce the complexity. Conversly active negotiation techniques can be used to

increase the cognitive complexity of the environment when there is conflict with another agent's solutions.

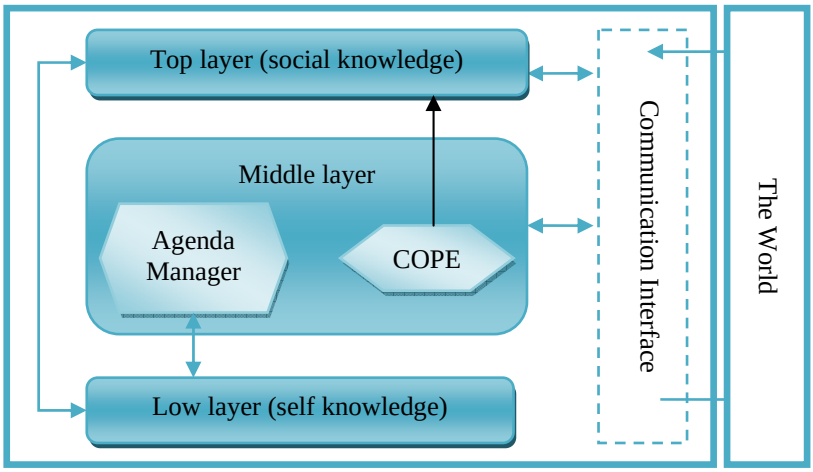


Figure 8.4 Agent structure in IMMUNE.

8.4.3 The Top Layer

This layer contains the social knowledge and is in charge of negotiation and coordination with other agents. It reports its information exchanges to the process blackboard of the CEO (Efatmaneshnik and Reidsema, 2009). The uppermost layer receives the potential solutions from the lowest layer and broadcasts them to other design agents that have interactions to the problem being considered. This layer also receives the same information from other agents and directly sends them to the lowest layer of the agent where the information is processed for determination of conflicts. The information about the conflicting objectives is also sent to the COPE module which determines the strategy of the negotiations (reactivity, pro-activity, selfishness) that is then reported to the upper most layer of the design agent (Efatmaneshnik and Reidsema, 2009).

8.5 Overall Behaviour

The decomposition module of the control source is in charge of decomposing the problem after the generation stage of a given abstraction level was accomplished The process of multi disciplinary team formation is based on the decomposition format of the problem, and SiFAs that were the elements of teams were grouped on this basis (Efatmaneshnik and Reidsema, 2009).

The control source of IMMUNE is active throughout the entire design process. The design process starts with the generation of an initial set of product variables upon a notification from the control shell to single agents to introduce their entries on the blackboard. This set of product variables act as a seed representing the highest abstraction hierarchy of the problem space. The seed might be a rough guess of what needs to be done (Figure 8.5). The simulation agent of the control source simulates the FL of the generated problem space and is in charge of gathering all the required information (for simulation) from the design agents.

Generally, this support system relies on and digests massive amount of information outcomes from the many simulations that might occur in each of the process layers. The decomposition agent of control source decomposes the set of generated variables and calls for the design agents' bids to participate in solving them. The decomposition module decides on one of the decomposition modes on the basis of estimated problem connectivity for a given abstraction level. The agents announce their interest back to the control shell by weighting their interest in solving each individual design variable or estimating the value of a design constraint. The composition agent assigns each individual parameter to a single design agent. In IMMUNE, the SiFAs are also grouped into virtual teams (coalitions). The composition agent announces the coalitions' formats according to one of the unsupervised integration modes described in Chapter 4. The IT manager is in charge of setting the shared mail boxes for each coalition, and also each layer.

The CEO agent of the control source estimates the minimum and maximum process cognitive complexities at all three levels of the coalitions, the layers and the entire system. The problem solving process starts at this stage; the problems are solved by the design agents and the results are sent back to the blackboard. During this process the information exchanges including those for conflict resolutions and negotiations in between the design agents take place entirely without the intervention of any module of the control source. CEO only monitors the design process cognitive complexity that is raised from the collaboration of the design agents during this last stage. If all the solutions are prepared the virtual groups are dismantled and collaboration process is stopped. The next set of the design variables are introduced and the cycle continues. It is also possible to start the next layer before the termination of the last stage.

The parallel execution of two or more layers of the problem solving process may need a single agent to be active in two or more process layers. The agenda manager of each individual design agent was a module that was introduced to maintain the focus of attention of the agent. The agents are allowed to introduce new product variables at any abstraction level at any time during the design process. They may also be cloned to solve two different design variable related to different design groups or in the same design group.

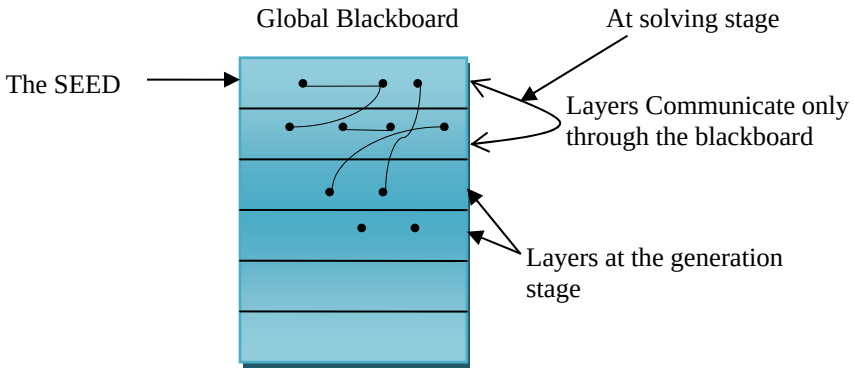


Figure 8.5 The design process at different abstraction levels may run simultaneously.

8.5.1 Adaptive Structuration in IMMUNE

As explained in Chapter 4 the problem solving modality in different abstraction levels can be disparate and based on the mode of decomposition at each level. We interpreted this *modality* as *adaptive structuration.* According to Chen and Li (2001), in order to enable dynamic structuration and adaptability, two preconditions must be satisfied:

1. The relevant design attributes, functions and variables necessary to formalize a design problem have been identified.
2. The interacting relation existing in teams is prescribed a priori in seeking a multi-team design solution.

In Chapter 7 we suggested that prior knowledge of the interactions (driven from the simulated PDSM) must remain as a suggestion and backbone to measure the process complexity, rather than to be used for forced communications through channels and filters. The reason to use this democratic method was to enhance the creativity across the entire design system and within the teams. The CEO module of the control source is in charge of measuring and tracking cognitive complexity of the integrated design system. However, the mode of problem decomposition and also integration mode remain unchanged for each layer after their introduction up until when the coalitions at that layer are dismantled.

To date, several other researchers have developed adaptable systems with modal functionalities. For example, Shen and Norrie (1998) argued that knowledge in modern manufacturing should have flexibility so that it can be applied to different kinds of applications. They used three different types of knowledge sharing patterns, primarily introduced by Tomiyama et al (1995) (Figure 8.6). There pattern were:

1. Independent knowledge bases: in this case, the strength of knowledge cannot be more than the sum of each of the independent knowledge bases (Figure 8.6(a)).

2. Integrated knowledge bases: Here, the knowledge bases can be applied to various situations and the strength of knowledge is near maximum (Figure 8.6(b)).
3. Interoperable knowledge bases: independent knowledge bases can communicate and form an interoperable situation (Figure 8.6(c)), although the *strength* of knowledge might be weaker than that in integrated knowledge bases.

Figure 8.6 Knowledge sharing architectures (Tomiyama et al 1995).

Another example was Zhang (1992) who classified types of problem solving among human experts in four predominant kinds according to their interdependencies:

1. *Horizontal cooperation* is where each expert in the cooperative group can get solutions to problems without depending on other experts, but if the experts cooperate, possibly using different expertise and data, they can increase confidence in their solutions.
2. *Tree cooperation* is where a senior expert depends on lower-level experts in order to get solutions to problems.
3. *Recursive cooperation* is where different experts mutually depend on each other in order to get solutions to problems.
4. *Hybrid cooperation* is where different experts use horizontal cooperation at some level in an overall tree or recursive cooperation.

Also Rosenman and Wang (1999) introduced the Component Agent-based Design-Oriented Model (CADOM) for collaborative design. This was a dynamic integrated model, using an integrated schema to contain data for multiple perspectives, but also with flexibility to support dynamic evolution. They recognized five types of modelling mechanisms for a collaborative design environment:

1. *Integrated mode* (Figure 8.7(a)): This is an integrated CAD system which works as a sharable server for all users using an integrated data model and a central management mechanism.
2. *Distributed-integrated mode* (Figure 8.7(b)): in this mode, distributed designers usually have their own domain systems along with a central service module called a sharable workspace.
3. *Discrete mode* (Figure 8.7(c)): this is a fully distributed system, where usually there is no central module but simply a set of distributed domain systems with discrete models and management mechanisms.
4. *Stage-based mode* (Figure 8.7(d)): In this mode a base model is set up at the first stage, and all subsequent models are derived from the base model.
5. *Autonomy-based mode* (Figure 8.7(e)): This is based on the concept of autonomy, in which each model is implemented as a distributed set of knowledge sources representing autonomous, interacting components.

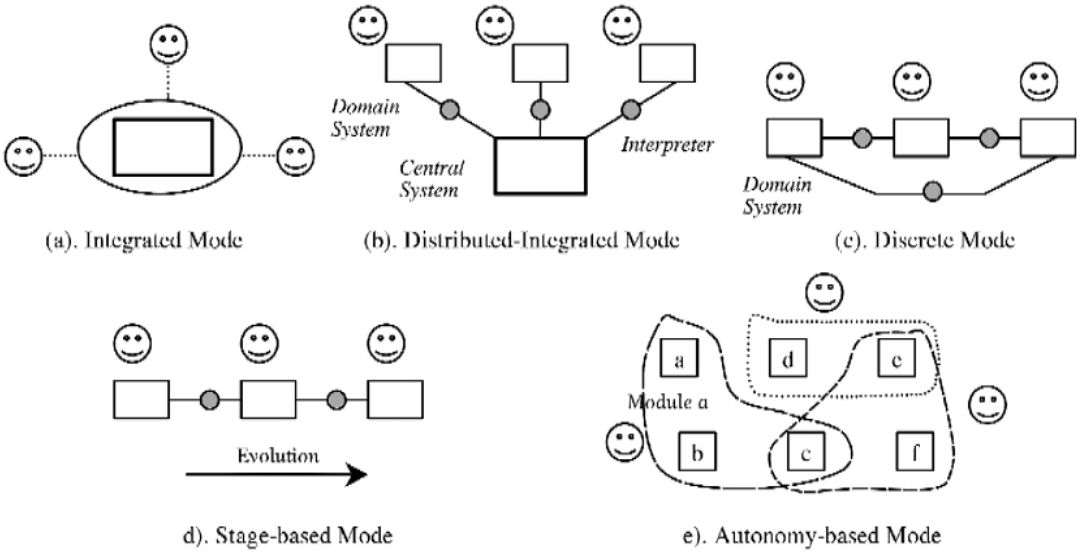


Figure 8.7 System modes for collaborative design systems (Rosenman and Wang, 1999).

We propose five modes of knowledge sharing and organizational structure that correspond to decomposition modes described in Chapter 4. These correspond to unsupervised integration schemes and are independent, integrative, multi agent, collaborative and competitive. It should be noted again that each process layer can acquire only one of these modes for the entire design process at that layer. Also it is important to note that the general structure of the system, here, is autonomy based and the following knowledge sharing patterns are highly and/or expected rather than definitive solid patterns. The process modes are described next (Efatmaneshnik and Reidsema, 2009).

8.5.2 Independent process mode

In this mode the decomposition agent has managed to fully decompose the problem; generally, very low self map connectivity that is the indication of a loose problem coupling leads to such situations. In this mode no collaboration is likely between the coalitions as depicted in Figure 8.8; this is so because when tasks are not interdependent, there is no need or reason to collaborate (Leenders et al, 2003). Consequently the need for radical innovation to integrate the system at the considered abstraction level would be minimal, and the process will be

characterized by short lead times. However collaboration exists between the design agents *inside* the same coalition. The CEO monitors the cognitive complexity inside the coalitions using the system knowledge provided by the agents regarding the degree of interaction with other members of the coalition; this is the only control relationship in the system.

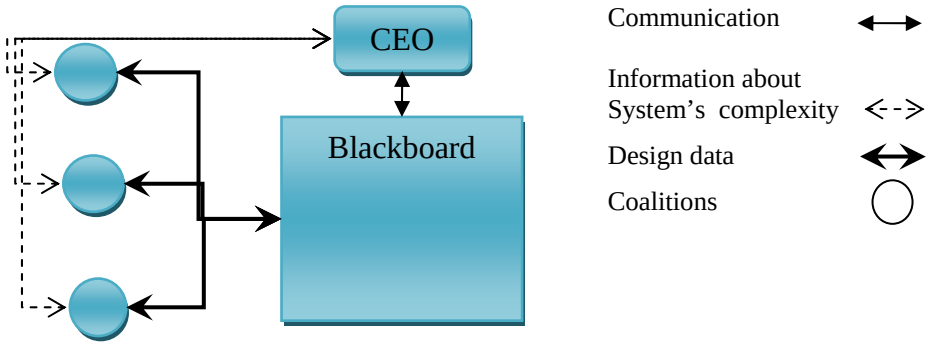


Figure 8.8 Independent process mode: coalitions do not need to communicate.

8.5.3 Integrative process mode

Integrative systems were explained in 4.5. Simple coordination of the design process makes this mode desirable, since all the coalitions have to coordinate their communications with only one integrative coalition. The corresponding (likely) organizational structure and integrative process mode is that illustrated in Figure 8.9.

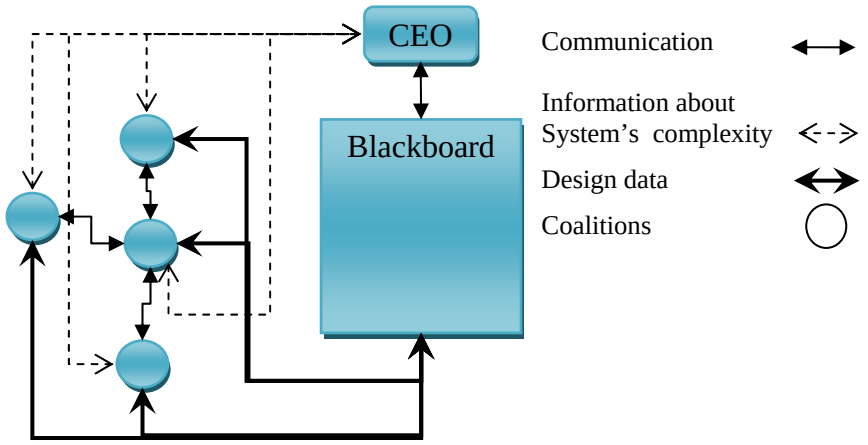


Figure 8.9 Integrative process mode. All coalitions exchange information with only one central coalition.

One drawback of this mode is that it might be hard for the design agents of the integrative coalition to maintain the cognitive complexity of the layer above the CEO-prescribed minimum cognitive complexity; since one module must be able to reach a high cognitive complexity. In other words the coordination in between several coalitions through one coalition might not be efficient or effective.

8.5.4 Autonomy based process mode

In this mode the agents explicitly act autonomously in their social behavior. The main characteristic of the autonomy process model is the intense cooperation amongst the design agents inside and across the boundaries of coalitions (Figure 8.10). This cooperation is an engine for innovation which is the main characteristic of autonomy based process.

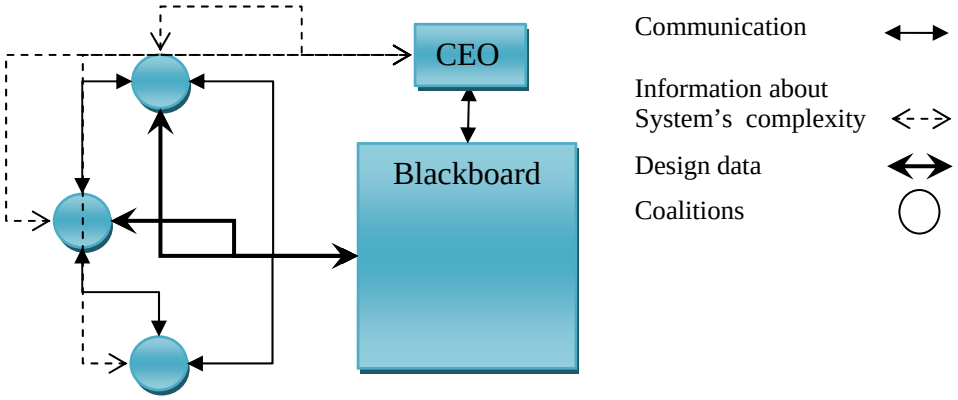


Figure 8.10 Autonomy based process model.

8.5.5 Collaborative process mode

In this mode subsystems are overlapped and they share some of the design variables with each other. As a result some of the coalitions explicitly share some of the agents, and there are some agents that have the membership of two or more coalitions (Figure 8.11). The real complexity is measured for overlap decompositions. As stated in 4.5 the main characteristic of this process mode is the intense collaboration between coalitions that makes this mode an information and knowledge intensive process (Klein et al, 2003.b).

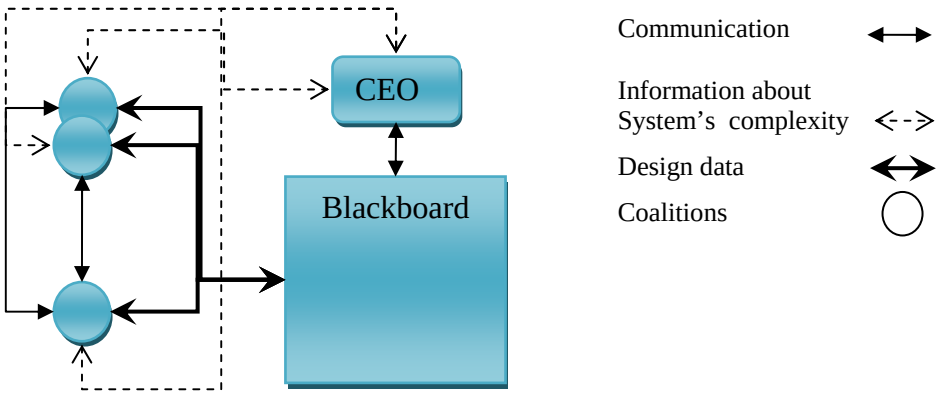


Figure 8.11 Collaborative process mode.

8.5.6 Competitive process mode

When the problems are very connected resulting in dense self maps, any kind of decomposition leads to large departures of the real complexity from the self complexity ($C_R >> C_S$) . In this condition, decomposition may not be a solution to problem tractability. In the competitive mode the problem is not decomposed, and each coalition tackles the entire problem by itself. The main characteristic of this process model is the competition between the design coalitions (Figure 8.12). The CEO monitors the cognitive complexity inside each coalition. Note that there naturally there shouldn't be any emergent system level cognitive complexity for this mode since it is assumed that coalitions do not no cooperate. However, informal cooperation may exist between the coalitions, but the cognitive complexity introduced by the informal cooperation of the coalitions can be ignored. The quality of the solutions is determined by the control source, based on the accuracy weights that the coalitions suggest for their solutions.

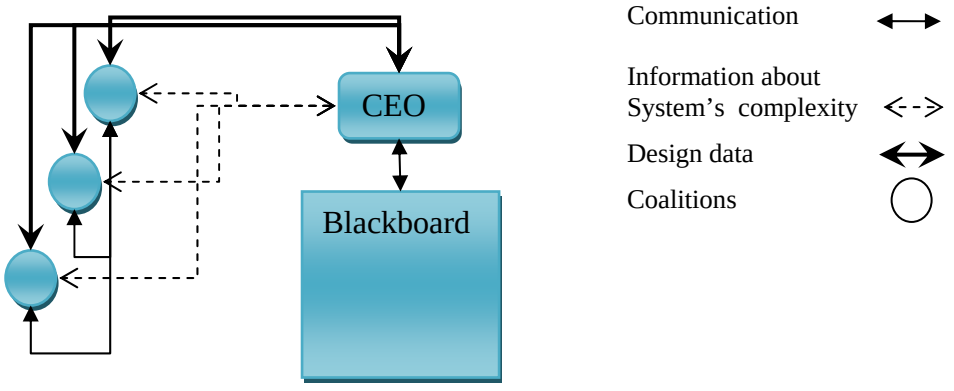


Figure 8.12 Competitive process mode, emulates the Enlightened Engineering design process.

9 Conclusion and Future Work

Complex engineered systems are large scale and densely coupled systems. This thesis consists of a design methodology, several design methods and design strategies for managing a complex product design; the design methods include process monitoring, modal decomposition, and immunity index for parametric decision making. The inherent fragility of a complex development process, a complex NPD organization and a complex product, are the central focus of these design methods. We used the term “immunity” to convey the message that complex systems must be immunized in order to be robust.

The presented design methodology was a model of distributed computation and complex problem solving as a new conceptual approach to concurrent parametric design of complex products. This approach extended the reach of parametric problem solving methodology to the conceptual stage, and by that addressed issues such as innovation and creativity that are subjects of conceptual design stage. This has been previously introduced by Kroll et al (2001) as the parameter analysis, and was regarded as an imperative for innovative conceptual design. The main contribution to the design methodology, here, was the inclusion of simulation based engineering to the gradualist model of design.

Design Gradualism asserts that a design problem must be introduced gradually and in steps usually regarded as abstraction levels. Abstraction levels refer to the levels of importance of the design problems for the main functionalities of the final product. One characteristic of complex products is the sensitivity of their core functionalities even to low level parameters (details) such as aesthetics. As an example consider the human skin; it can be regarded as a detail (thus low level) in the abstraction hierarchy, although it plays sensitive roles in overall health and immunity of the body. Therefore the argument is that for complex products the details and the core functionalities must be developed and designed together. As such we made the comparison between the first set of design variables and the seed of a plant. The seed needs to have a foretaste of all abstractions in the product (functionalities). The termination criteria for each

design stage (abstraction level) were also introduced as the number of design variables.

The presented template suggests simulation as a tool for the purpose of design management as well as its commonly accepted purpose: an imperative for design optimization and performance improvement. Monte Carlo Simulation and the Global Entropy Based Correlation Coefficients were suggested for establishing, early in the design phase, the self map of the system which shows the sensitivity of design objectives and variables. This self map is represented as a weighted graph or parametric based design structure matrix (Browning, 2001). Decomposition is a design method, which is applicable to the design problem (self map). Modal decomposition of the problem space was suggested: a problem may be decomposed in several modes depending on the connectivity (or coupling) of the design variables at each level of abstraction. These decomposition modes are described as being analogous to the growing connectivity of the problem and are defined as: 1) Full decomposition, all subsystems (sub-problems) are independent for least connected systems. 2) Integrative or coordination based decomposition; where one subsystem (named integrative subsystem) is connected to all other independent subsystems that are independent. 3) Modular or multi agent decomposition, where all subsystems or some of them are connected 4) Overlap decomposition, which is similar to Multi Agent decomposition with the exception that some of the subsystems are overlapped indicating shared design and objective variables. 5) No decomposition for densely connected systems that show strong emergence.

We also presented three design methods that were based on the measures of complexity of the design problems, cognitive complexity of the problem solvers and complexity of the design solutions. Here, we referred to the problem structure as the self map of the problem or the under-development system. If the problem is large the self needs to be decomposed for tractability. It was noted that the structure of the problem after decomposition is the real structure to be dealt with. We referred to the complexity of the system/problem before decomposition as the self complexity and complexity of system after decomposition as the real complexity. It was shown that the real complexity cannot be less than the self

complexity, thus, contrary to what is commonly assumed, decomposition does not reduce the complexity. The immune decomposition was introduced that utilized real complexity as the quality partitioning criteria which can be applied to all types of decompositions: integrative decompositions, modular decompositions as well as overlap decompositions. The real complexity enables comparison between the outcomes of these three decompositions. It was reasoned that minimizing the real complexity leads to better efficiency and effectiveness of the design process. Real complexity constitutes the approach of this thesis to immunization of the design process.

Cognitive complexity was defined for a design organization. Cognitive complexity is the ability of a person or an organization to integrate a system. We suggested measuring cognitive complexity of an organization as a function of information exchanges between design agents and in general design units (e.g. design teams) by applying our graph theoretic complexity measure to the monitored team based DSM. In order to integrate and manage a complex problem the problem solving system requires a cognitive complexity that is more, or at least equal, to the complexity of the problem. Our method to immunization of complex design organizations constituted maintaining the cognitive complexity of an organization around the complexity of the problem (in between the minimum and maximum complexity of the problem). It was also noted that for hierarchical organizations e.g. federation of coalitions and collection of federations the cognitive complexity and their bounds must be measured as the real cognitive complexity of the hierarchies.

Finally, Immunity Index was introduced to insure the immunity from sudden collapse at the product level. It was based on the It was noted that the variation in some parameters of a system can affect the underlying structural relationship between the system's variables. Based on this notion, a globally robust product was regarded as one that uncertainties in the environment as the variation in the product's input variables were unable to change the structural properties of the product's variables. This way, the unexpected behaviour is more unlikely, since a source of failure is the modal bifurcation of the structures.

A conceptual DSS was introduced that incorporated the design methodology, design methods and strategies into a flat organizational architecture. IMMUNE is capable of adaptive structuration which is planning decisions in a metamorphose environment for each of the five mentioned decomposition modes. Design agents are clustered within each GDDI cycle as virtual teams or coalitions of agents whose structure mimics the structure of the problem. Subsequently, and correspondingly to the five modes of decomposition, the IMMUNE architecture is capable of employing five modes of design integration: 1) Independent mode that is fully concurrent problem solving. 2) Integrative mode which is coordination based problem solving. 3) Autonomy based problem solving that is cooperative and on the basis of cooperation of several coalitions of agents. 4) Collaborative problem solving where some of the coalitions of agents are semi merged and overlapped 5) Competitive problem solving on basis of Enlightened Engineering in which several independent coalitions of agents competing to solve the same problem.

Adaptive Structuration is accomplished by employing a global blackboard containing the current state of the design at all abstraction levels. The control source decides on the decomposition mode based on the connectivity of the problem and then decomposes it on the basis of minimum real complexity. The CEO module of the control source is in charge of maintaining the coherence of the multi agent design environment. CEO informs all the design agents of the amount of cognitive complexity of their coalition and the federation. The COPE module of the design agents are then in charge of maintaining the cognitive complexity of the coalitions and the federation above the announced (by CEO) minimum and away from the maximum bound. COPE module, decides on the high level interactions mode (passive or proactive-social) by using the conflict resolution strategies that are passive like constraint relaxation or proactive such as active negotiation.

The presented architecture is IMMUNE against sudden failure in meeting the top level organization objectives including cost, lead time and the quality of the product. It is often argued that complex systems are robust yet in the presence of uncertainties they become fragile; this strange behaviour is related to the

chaotic and sensitive characteristic of complex systems. In the domain of sustainability of the organizations that design complex products this means that the top level goals may often be robustly met, however, sudden and large departures from those goals may seem inevitable. To immunize against this fragility the proposed system advocates coherency in collaboration. That is, the locally aware design agents (aware of their local tasks) maintain the global coherency, harmony and order through their COPE module by making the agents' social behaviour subservient to the information the system's cognitive complexity received from the CEO module.

The immunity discussed in this thesis is more of a "*reasoning by metaphor*" which means that there has not been much enough effort to study the natural immune systems characteristics. One reason for this was that the limited timing of the PhD (4 years period) was only enough to introduce myself to the field and notions of complex systems science. A more rigorous approach could be to use detail description of the recently uncovered as well as new understandings of the already known mechanisms of natural immune system. These mechanisms should be reflected on and incorporated into a distributed computation strategies; and by that the core output of the research would be immune computational strategies within the context of machine learning. The incorporation of complexity measure into artificial immune systems and algorithms, however, is a novel approach that has not been tried yet. The implementation of IMMUNE by an Artificial Intelligence programming language such as LISP is an important continuation of this project.

References

Allen MS. 1952. *Morphological Creativity.* Prentice-Hall, Englewood Cliffs, NJ.

Alexander C. 1964. *Notes on the Synthesis of Form*. Harvard University Press, Cambridge.

Alpert CJ, Kahng AB, and Yao SZ.1999. Spectral Partitioning with Multiple Eigenvectors. *Discrete Applied Mathematics*, Vol 90, pp 3-26.

Altus SS, Kroo IM, and Gage J. 1996. A Genetic Algorithm for Scheduling and Decomposition of Multidisciplinary Design Problems. *Journal of mechanical design*, Vol 118, No 4, pp 486-89.

Alstyne V, and Logan GR. 2007. Designing for Emergence and Innovation: Redesigning Design. *Artifact*, 2007, Vol 1, No 2, pp 120-129.

Anderson C. 2006. Creation of Desirable Complexity: Strategies for Designing Self-Organized Systems. In *Complex Engineered Systems*, Eds Braha, Minai and Bar-Yam, pp 22-39. Springer, Cambridge, Massachusetts.

Arbib MA, and Manes EG. 1974. Foundations of System Theory: Decomposable Systems. *Automatica*, Vol 10, pp 285-302.

Ashby WR, and Gardner M. 1970. Connectance of Large, Dynamic Cybernetic Systems: Critical Values for Stability. *Nature*, pp 228: 784.

Balazs M, and Brown D. 2002. Design Simplification by Analogical Reasoning. In *From Knowledge Intensive CAD to Knowledge Intensive Engineering.* Eds Cugini and Wozny, pp 29-44, Kluwer Academic Press, Netherlands.

Banzhaf W. 1994. Self-Replicating Sequences of Binary Numbers: The Build-up of Complexity. *Complex Systems*, Vol 8, pp 215-225.

Bar-Yam Y. 2002. *Large Scale Engineering and Evolutionary Change:*

Useful Concepts for Implementation of FORCEnet. Report to Chief of Naval Operations Strategic Studies Group.

Bar-Yam Y. 2003. *When systems engineering fails - toward complex systems engineering.* In proceedings of IEEE International Conference on Systems, Man, and Cybernetics, No 2, pp 2021- 2028.

Bar-Yam Y. 2004. *Making Things Work: Solving Complex Problems in a Complex World.* NECSI Knowledge Press, Cambridge.

Bashir HA, and Thomson V. 1999.a. Metrics for design projects: a review. *Design Studies,* Vol 20, pp 263–277.

Bashir HA, and Thomson V. 1999.b. Estimating Design Complexity. *Journal of Engineering Design,* Vol 10, No 3.

Bayrak C, and Tanik MM. 1998. A Process Oriented Monitoring Framework. *Systems Integration,* Vol 8, pp 53-82.

Bechir A, and Kaminska B. 1995. CYCLOGEN: Automatic, Functional-Level Test Generator Based on the Cyclomatic Complexity Measure and on the ROBDD Representation. *IEEE Transactions on Circuits and Systems - II: Analogue and Digital Signal Processing,* Vol 42, pp 446-452.

Boehm B. 1988. A Spiral Model of Software Development and Enhancement. *IEEE Computer*, Vol 5, pp 61-72.

Boschetti F, Prokopenko M, Macreadie I, et al. 2005. *Defining and Detecting Emergence in Complex Networks.* In proceedings of 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, Melbourne, Australia, September 14-16, p573:580.

Braha D, and Bar-Yam Y. 2006. The Structure and Dynamics of Complex Product Design. In *Complex Engineered Systems: Science Meets Technology,* Eds Minai A, Braha D, Bar-Yam Y, pp 40-71, Springer, Cambridge, Massachusetts.

Braha D, and Maimon O. 1998. The Measurement of a Design Structural and Functional Complexity. *IEEE Transactions on Systems, Man, and Cybernetics-Part A,* Vol 28, No 4, pp 527-535.

Brown DC, Dunskus B, Grecu DL et al. 1995. *SINE: Support for single function agents*. Paper presented in Applications of AI in Engineering, Udine, Italy.

Browning TR. 1999. Designing system development projects for organizational integration. *Systems Engineering,* Vol 2, pp 217-225.

Browning TR. 2001. Applying the Design Structure Matrix to System Decomposition and Integration Problems: A Review and New Directions. *IEEE Transactions on Engineering Management,* Vol 48, No 3, pp 292-306.

Browning TR, Deyst J, and Eppinger S D. 2002. Adding Value in Product Development by Creating Information and Reducing Risk. *IEEE Transactions on Engineering Management*, Vol 49, No 4, pp 443-58.

Cameron PJ. 2004. Automorphisms of Graphs. In *Topics in Algebraic Graph Theory*, Eds Beineke LW, and Wilson RJ, Cambridge University Press, pp 137-55.

Carneiro RL. 1987. The Evolution of Complexity in Human Societies and its Mathematical Expression. *International Journal of Comparative Sociology*, Vol 28, pp 111-128.

Caro S, Bennis F, and Wenger P. 2005. Tolerance Synthesis of Mechanisms: A Robust Design Approach. *Journal of Mechanical Design*, Vol 127, No 1, pp 86–94.

Carver N, and Lesser V. 1994. The Evolution of Blackboard Control Architectures. *Expert Systems with Applications*, Vol 7, pp 1-30.

Casti JL. 1977. *Complexity, Connectivity and Resilience in Complex Ecosystems.* IFAC Symposium on Bio- and Ecosystems, Liepzig, Germany.

Casti JL. 1994. *Complexification: Explaining a Paradoxical World Through the Science of Surprise*. Abacus, London.

Chan PK, Schlag MD, and Zien JY. 1994. Spectral K-Way Ratio-Cut Partitioning and Clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol 13, No 9, pp 1088 – 1096.

Chen L, Ding Z, Simon L. 2005. A Formal Two-Phase Method for Decomposition of Complex Design Problems. *Journal of mechanical design*, Vol 127, pp 184-195.

Chen L, and Li S. 2001. *Concurrent Parametric Design Using a Multifunctional Team Approach.* Paper presented in Design Engineering Technical Conferences DETC‘01. Pittsburgh, Pennsylvania.

Chen L, and Li S. 2005. Analysis of Decomposability and Complexity for Design Problems in the Context of Decomposition. *Journal of Mechanical Design*, Vol 127, pp 545-557.

Cheng CK, and Hu TC. 1989. *The optimal partitioning of networks*. Technical Report, University of California, San Diego.

Chiva-Gomez R. 2004. Repercussions of complex adaptive systems on product design management. *Technovation*, Vol 24, pp 707-711.

Cisse A, Ndiaye S, and Link-Pezet J. 1996. *Process Oriented Cooperative Work: an Emergent Framework.* In IEEE Symposium and Workshop on Engineering of Computer Based Systems, pp 342-347, Friedrichshafen, Germany.

Clark KB, and Fujimoto T. 1989. *Overlapping Problem Solving in Product Development.* In proceeding of Managing International Manufacturing Conference, pp 127-152, Amsterdam, North-Holand.

Cohen I. 2007. Real and Artificial Immune Systems: Computing the State of the Body. *Immunological Reviews,* Vol 7, pp569-574.

Conant RC. 1972. Detecting Subsystems of a Complex System. *IEEE Transactions on Systems,Man and Cybernetics*, Vol 2, pp 550-553.

Cook R. 2000. *How complex systems fail*. Cognitive Technologies Laboratory Publication, University of Chicago. Chicago, IL. http://www.ctlab.org/documents/How%20Complex%20Systems%20Fail.pdf

Cross N. 2000. *Engineering Design Methods: Strategies for Product Design.* Third Edition, John Wiley and Sons Ltd.

Corkill DD. 1991. Blackboard Systems. *AI Expert*, Vol 6, pp 40-47.

Corkill DD. 2003. *Collaborating Software: Blackboard and Multi-Agent Systems & the Future.* Paper presented in International Lisp Conference, New York.

Craig ID. 1993. *Formal Techniques in the Development of Blackboard Systems.* Research Report Coventry, Department of Computer Science, University of Warwick, UK.

Crutchfield JP. 1994. The calculi of emergence: Computation, dynamics and induction. *Physica D*, Vol 75, pp 11–54.

Crutchfield JP, and Young K. 1898. Inferring Statistical Complexity. *Physical Review Letters*, Vol 63, pp 105-108.

Curtis B, Sheppard SB, and Milliman P. 1979. Measuring the psychological complexity of software maintenance tasks with the Halstead and McCabe metrics. *IEEE Transactions on Software Engineering*, Vol 5, pp 96-104.

Darbellay G. 1998. Predictability: an Information-Theoretic Perspective. In *Signal Analysis and Prediction*, Eds Proch´azka A, Uhl´ır J, Rayner PJW, and Kingsbury NG, pp 249-262, Boston.

Dasgupta D. 1998. *An Artificial Immune System as a Multi-Agent Decision Support System*. In proceedings of IEEE International Conference on Systems,

Man and Cybernetics (SMC), October , Vol 4, pp 3816-3820, San Diego, California.

Dembski W. 2002. *No Free Lunch: Why Specified Complexity Cannot Be Purchased without Intelligence*. Rowman & Littlefield Publishers, Inc.

DeSanctis G, and Monge P. 1999. Communication processes for virtual organizations. *Organization Science*, Vol 10, pp 693-703.

DeSanctis G, and Poole MS. 1994. Capturing the Complexity in Advanced Technology Use: Adaptive Structuration Theory. *Organization Science*, Vol5, pp 121-147.

Dierneder S, and Scheidl R. 2001. Complexity Analysis of Systems from a Functional and Technical Viewpoint. *Lecture Notes in Computer Science LNCS* No 2178, pp 223-232, Springer-Verlag Berlin Heidelberg.

Diestel R. 2005. *Graph Theory*. Third ed, Springer-Verlag, Heidelber/ New York.

Ding C, He X, Zha H, et al. 2001. *A Min-Max Cut Algorithm for Graph Partitioning and Data Clustering*. IEEE International Conference on Data Mining, Dec. 2, San Jose, CA.

Dionisio A, Menezes R, and Mendes DA. 2007. Entropy and Uncertainty Analysis in Financial Markets. *ArXiv e-print*, arXiv:0709.0668, http://adsabs.harvard.edu/abs/2007arXiv0709.0668D.

Dissanayake K, and Takahashi M. 2006. The Construction of Organizational Structure: Connections with Autopoietic Systems Theory. *Contemporary Management Research*, Vol 2, pp 105-116.

Druzdzel MJ, and Flynn RR. 2000. Decision Support Systems. In *Library and Information Science*, Ed Kent A, pp 120-133, Marcel Dekker Inc, New York.

Duin RPW, and Pekalska E. 2006. Object Representation, Sample Size and Dataset Complexity, In *Data Complexity in Pattern Recognition*, Eds Basu M, and Ho TK, Springer, pp. 25-47.

Dunskus BV. 1994. *Single Function Agents and Their Negotiation Behavior in Expert Systems.* Research Report, Worcester Polytechnic Institute, Worcester, MA.

Dussauchoy RL. 1982. Generalized Information theory and the Decomposability of Systems. *International Journal of General Systems*, Vol 9, pp 13-36.

Edmonds B. 1997. *Hypertext Bibliography of Measures of Complexity*. http://www.cpm.mmu.ac.uk/_bruce/combib/

Edmonds B. 1999. *Syntactic Measures of Complexity*. PhD Thesis, University of Manchester, Manchester, UK.

Efatmaneshnik M, and Reidsema CA. 2007.a. Immunity as a Design Decision Making Paradigm for Complex Systems: a Robustness Approach. *Cybernetics and Systems*, Vol 38, No 8, pp 759-780.

Efatmaneshnik M, and Reidsema CA. 2007.b. *Immunity and Information Sensitivity of Complex Product Design Process in Overlap Decomposition.* In Proceedings of 7th International Conference on Complex Systems, Boston, MA.

Efatmaneshnik M, and Reidsema CA. 2008.a. *Decomposition Modes and Integration Schemes in Complex Systems Design.* In proceedings of Systems Engineering Test and Evaluation SETE 2008, September, Canberra.

Efatmaneshnik M, and Reidsema CA. 2008.b. *Exploiting Non-Dominance in Multi Agent Systems: An Artificial Immune Algorithm for Distributed and Complex Problem Solving Environments.* In proceedings of 12th Asia Pacific Symposium on Intelligent and Evolutionary Systems IES08, 7-8 December, Melbourne.

Efatmaneshnik M, and Reidsema CA. 2009. IMMUNE: A Collaborating Environment for Complex System Design. In *Studies in Computational Intelligence: Collaboration, Fusion and Emergence*, Eds Mumford C, and Jain L, Ch 9, Springer. "in press"

El-Haik B, and Yang K. 1999. The components of complexity in engineering design. *IIE Transactions*, Vol 31, No 10, pp 925-934.

English JR, and Taylor GD. 1993. Process Capability Analysis: A Robustness Study. *International Journal of Production Research*, Vol 31, No 7, pp 1621–1635.

Eppinger, SD. 1997. *A Planning Method for Integration of Large Scale Enginering Systems.* Paper presented at the International Conference on Engineering Design ICED 97, Tampere, Finland, August 19-21.

Eppinger, SD, and Salminen V. 2001. *Patterns of Product Development Interactions.* Paper presented at the International Conference on Engineering Design, Glasgow, Scotland, August, 21-23.

Eppinger SD, Whitney DE, and Gebala DA. 1992. *Organizing the Tasks in Complex Design Projects: Development of Tools to Represent Design Procedures.* In NSF Design and Manufacturing Systems Conference, pp 301-309, Atlanta, Georgia.

Eppinger SD, Whitney D, Yassine A, et al. 2003. Information Hiding in Product Development: The Design Churn Effect. *Research in Engineering Design*, Vol 14, No 3, pp 145-161.

Erdi P. 2008. *Complexity Explained.* Springer-Verlag.

Fathi Y, and Palko D. 2001. A Mathematical Model and Heuristic Procedure for the Robust Design Problem with High-Low Tolerances. *IIE Transactions*, Vol 33, No 12, pp 1121–1127.

Feng CX, and Balusu R. 2002. Robust Tolerance Design Considering Progress Capability and Quality Loss. *Conceptual and Innovative Design for Manufacturing*, Vol 103, pp 1–14.

Finger S, and Dixon JR. 1989. A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive, and Computer-Based Models of Design Processes. *Research in Engineering Design*, No 1, pp 51-67.

Fixson SK. 2005. Product architecture assessment: a tool to link product, process, and supply chain design decisions. *Journal of Operations Management*, Vol 23, No 3-4, pp 345-369.

Formica A, Marczyk J. 2007. *Strategic Multiscale A New Frontier for R&D and Engineering.* Ontonix, Turin.
http://www.ontonix.com/index.php?page=download&CID=36

Fraser AM. 1989. Measuring Complexity in Terms of Mutual Information. In *Measures of Complexity and Chaos*, Ed Abraham NB, pp 117-119, Plenum Press, New York.

French MJ. 1985. *Conceptual Design for Engineers*. Design Council, London.

Fruchter R, Clayton MJ, Krawinkler H, et al. 1996. Interdisciplinary Communication medium for Collaborative Conceptual Building Design. *Advances in Engineering Software*, Vol 25, pp 89-101.

Fyfe C, and Jain L. 2006. Teams of intelligent agents which learn using artificial immune systems. *Journal of Network and Computer Applications*, Vol 29, pp 147–159.

Gero J S. 1996. Creativity, emergence and evolution in design. *Knowledge-Based Systems*, Vol 9, pp 435-448.

Ghanea-Hercock R. 2007. Survival in cyberspace. *Information Security*, Vol, 12, pp 200–208.

Gilb T. 1988. *Principles of Software Engineering Management*. Reading Addison-Wesley Publishing Company, MA.

Goel S, and Gangolly J. 2007. On decision support for distributed systems protection: A perspective based on the human immune response system and epidemiology International. *Journal of Information Management*, Vol 27, pp 266–278.

Gorodkin J, Sorensen A, and Winther O. 1993. Neural Networks and Cellular Automata Complexity. *Complex Systems*, Vol 7, pp 1-23.

Grassberger P. 1989. Problems in Quantifying Self-organized complexity. *Helvetica Physica Acta*, Vol 62, pp 498-511.

Griffin A. 1993. Metrics for Measuring New Product Development Cycle Time. *Journal of Product Innovation Management*, No 10, pp 112–125.

Gulati RK, and Eppinger SD. 1996. *The Coupling of Product Architecture and Organizational Structure Decisions.* MIT publication, Cambridge, MA.

Hamilton DP. 2001. Circuit Break: Gambling It Can Move Beyond the PC, Intel Offers a New Microprocessor. *Wall Street Journal*, No 29, May issue.

Hart E, McEwan C, and Davoudani D. 2009. Exploiting Collaborations in the Immune Systems: the future of artificial immune systems. In *Studies in Computational Intelligence: Collaboration, Fusion and Emergence*, Eds Chrisitne Mumford and Lakhmi Jain, Ch 16, Springer. 2009

Henderson RM, Clark KB. 1990. Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms. *Administrative Science Quarterly*, Vol 35, No1, pp 9-30.

Hinds P, and McGrath C. 2006. *Structures that work: social structure, work structure and coordination ease in geographically distributed teams.* In Proceedings of 2006 the 20th anniversary conference on Computer supported cooperative work (CSCW '06), Nov 04-08, pp 343-352, Banff, Alberta, Canada.

Hobday M, Rush H, and Tidd J. 2000. Innovation in Complex Products and System. *Research Policy*, Vol 29, pp 793-804.

Hollingsworth P, and Mavris DN. 2000. *A Method for Concept Exploration of Hypersonic Vehicles in the Presence of Open and Evolving Requirements.* Paper presented at World Aviation Conference October 10-12, San Diego, CA.

Hops JM and Sherif JS. 1995. Development and application of composite complexity models and a relative complexity metric in a software maintenance environment. *Journal of Systems and Software,* Vol 31, pp 157-169.

Iansiti JC. 1990. Microsoft Corporation: Office Business Unit. *HBS Case*, Vol 9, pp 691-330.

Ikeda M, Siljak DD, and White DE. 1981. Decentralized Control with - Overlapping Information Sets. *Journal of Optimization Theory and Applications*, Vol 34, No 2, pp279-310.

Kan J, and Gero J. 2005. *Can entropy Represent Design Richness in Team Designing?* In pproceedings of the 10th International Conference on Computer Aided Architectural Design Research in Asia CAADRIA'05, Ed Bhatt A, New Delhi, pp 451-457.

Kannapan SM. 1995. Function Metrics for Engineered Devices. *Applied Artificial Intelligence*, Vol 9, No 1, pp 45-64.

Keating C, Rogers R, Unal R, et al. 2003. System of Systems Engineering. *Engineering Management Journal*, Vol 15, No 3, pp 36-45.

Kemper S, Rice K, and Chen Y. 1995. Complexity metrics and growth curves for measuring grammatical development from five to ten. *First Language*, Vol 15, pp 151-166.

Kindlmann P. 1984. Stability vs. Complexity in Model Computational Communities. *Lecture Notes in BioMathematics*, Vol 54, pp 191-207.

Klein M, Braha D, Syama H, and Bar-Yam Y. 2003.a. Editorial: Special Issue on a Complex Systems Perspective on Concurrent Engineering. *Concurrent Engineering Research and Applications*, Vol 11, No 3, pp 163.

Klein M, Sayama H, Faratin P, and Bar-Yam Y. 2003.b. The Dynamics of Collaborative Design: Insights from Complex Systems and Negotiation Research. *Concurrent Engineering Research & Applications*, Vol 11, No 3, pp 201-209.

Klir, GJ. 2003. Facets of Generalized Uncertainty-Based Information. In *Entropy Measures, Maximum Entropy Principle and Emerging Applications*, Ed Karmeshu J, Springer, pp 55-75.

Ko KH, Pochiraju K, and Manoochehri S. 2007. Dynamic Evolution of Information Complexity for Analysis of Design and Development. *Journal of Advanced Mechanical Design, Systems, and Manufacturing*. Vol 1, No 1, pp 36-47.

Krishnan V, Eppinger SD, and Whitney DE. 1997. A Model-Based Framework to Overlap Product Development Activities. *Management Science*, Vol 43, No 4, pp 437-51.

Kratzer J, Leenders RTAJ, and Engelen JMLV. 2004. A Delicate Managerial Challenge: How Cooperation and Integration Affect the Performance of New Product Development teams. *Team Performance Management*, Vol 10, pp 20-25.

Kroll NEA, and Klimesch W. 1992. Semantic Memory - Complexity or Connectivity. *Memory and Cognition*, Vol 20, pp 192-210.

Kroll E, Condoor SS, and Jansson DG. 2001. *Innovative Conceptual Design*. Cambridge University Press.

Kuras ML. 2007. An Introduction to Complex-System Engineering, *InterJournal Complex Systems*, manuscript ID 2006.

Kurtoglu T. 2007. *A Computational Approach to Innovative Conceptual Design*. PhD Thesis, University of Texas at Austin.

Kusiak A. 1999. *Engineering Design: Products, Processes, and Systems*. Academic Press.

Kusiak A, and Feng CX. 2000. Robust Tolerance Synthesis with the Design of Experiments Approach. *Journal of Manufacturing Science and Engineering*, Vol 122, No 3, pp 520–528.

Lakshmanan KB, Jayaprakash S, and Sinha PK. 1991. Properties of Control-Flow Complexity Measures. *IEEE Transactions on Software Engineering*, Vol 17, pp 1289-1295.

Lander SE, Staley SM, and Corkill DD. 1996. Designing Integrated Engineering Environments: Blackboard-Based Integration of Design and Analysis Tools. *Concurrent Engineering: Research and Applications*, Vol 4, pp 59-72.

Larson J R. 2007. Deep Diversity and Strong Synergy: Modeling the Impact of Variability in Members' Problem-Solving Strategies on Group Problem-Solving Performance. *Small Group Research*, Vol 38, pp 413-436.

Lau H, and Wong V. 2004. Immunologic Responses Manipulation of AIS Agents. *Lecture Notes in Computer Science*, Vol 3239, pp 65-79.

Lazarev VI. 1992. Complexity and Synthesis of Minimal Logic Circuits using Multiplexers. *Cybernetics*, Vol 28, pp 796-799.

Lee J, and Truex DP. 2000. *Cognitive Complexity and Methodical Training: Enhancing or Suppressing Creativity*. In proceedings of 33rd International Conference on System Sciences, Hawaii.

Lee T. 2003. *Complexity Theory in Axiomatic Design.* PhD Thesis Massachusetts Institute of Technology. Dept. of Mechanical Engineering.

Leenders RTAJ, Kratzer J, and Hollander J. 2003. Virtuality, Communication, and New Product team Creativity: a Social Network Perspective. *Engineering and Technology Management*, Vol 20, No 1, pp 69-92.

Lissack M. 1999. Complexity: the Science, its Vocabulary, and its Relation to Organizations. *Emergence*, Vol 1, No 1, pp 110-126.

Liu H, and Chen W. 2006. Relative Entropy Based Method for Probabilistic Sensitivity Analysis in Engineering Design. *Journal of Mechanical Design*, Vol 128, No 2, pp 326–336.

Liu Y C, Chakrabarti A, and Bligh T P. 2003. Towards an Ideal Approach for Concept Generation. *Design Studies Journal*, Vol 24, No 4, pp 341-355.

Marczyk J. 1999. *Principles of Simulation Based Computer Aided Engineering.* FIM Publications, Barcelona.

Marczyk J. 2002. *Beyond optimization in computer-aided engineering.* International Centre for Numerical Methods in Engineering, Barcelona.

Marczyk J, and Deshpande B. 2006. *Measuring and Tracking Complexity in Science.* In proceedings of 6th International Conference on Complex Systems, Boston, MA.

Marczyk J. 2008. Complexity Management : New Perspective and Challenges for CAE in the 21-st Century. *BENCHmark Magazine*, July issue, pp 13-17.

Margalef R. 1984. Ecosystems: Diversity and Connectivity as measurable components of their complication. In *The Science and Praxis of Complexity*, Ed Aida et al, pp 228-244, United Nations University, Tokyo.

Maturana F, Shen W, and Norrie DH. 1999. MetaMorph: an Adaptive Agent-Based Achitecture for Intelligent Manufacturing. *International Journal of Production Research*, Vol 37, pp 2159 – 2173.

McCabe TJ. 1976. A Complexity Measure. *IEEE Transactions on Software Engineering*, Vol 2, No 4, pp 308-320.

Mcconnell S. 1996. *Rapid Development: Taming Wild Software Schedules*. Microsoft Press, Redmond.

McDonald RA, and Mavris DN. 2000. *Formulation, Realization, and Demonstration of a Process to Generate Aerodynamic Metamodels for Hypersonic Cruise Vehicle Design.* Paper presented in World Aviation Conference, San Diego, CA, October 10-12.

Meinel C. 1990. Logic vs. Complexity Theoretic Properties of the Graph Accessibility Problem for Directed Graphs of Bounded Degree. *Information Processing Letters*, Vol 34, pp 143-146.

Mellacheruvu PV, Fu MC, and Herrmann JW. 2000. *Comparing Gradient Estimation Methods Applied to Stochastic Manufacturing System.* Technical Report, Institute for systems Research, University of Maryland.

Merry U. 1995. *Coping with Uncertainty: Insights from the New Sciences of Chaos, Self-Organization, and Complexity*. Praeger, London.

Meunier K, and Dixon JR. 1988. *Iterative Respecification: A Computational Model for Hierarchical Mechanical System Design.* In proceedings of the ASME Computers in Engineering Conference, American Society of Mechanical Engineers, San Francisco, CA, July 31-August 3.

Michelena NF, and Papalambros PY. 1995. A Network Reliability Approach to Optimal Decomposition of Design Problems. *Journal of Mechanical Design*, Vol 117, No 3, pp 433-40.

Michelena NF, and Papalambros PY. 1997. A Hypergraph Framework for Optimal Model-Based Decomposition of Design Problems. *Computational Optimization and Applications*, Vol 8, pp 173-96.

Mihm J, and Loch CH. 2006. Spiraling out of Control: Problem-Solving Dynamics in Complex Distributed Engineering Projects. In *Complex Engineered Systems*, Eds Braha, Minai and Bar-Yam, pp 141-157, Springer, Cambridge, Massachusetts.

Minati M, and Pessa L. 2006. *Collective beings: Contemporary Systems Thinking*. Springer, New York.

Minai AA, Braha D, and Bar-Yam Y. 2006. Complex Engineered Systems: A New Paradigm. In *Complex Engineered Systems: Science Meets Technology*, Eds Minai AA, Braha D, Bar-Yam Y, pp 1-21, Springer, Cambridge, Massachusetts.

Monceyron E, and Barthes JP. 1992. Architecture for ICAD Systems: an Example from Harbor Design. *Sience et Techniques de la Conception*, No 1, pp 49-68.

Naylor AW. 1981. On Decomposition Theory: Generalised Dependence. *IEEE Transactions on Systems, Man and Cybernetics*, Vol 10, pp 699-713.

Norman DO, and Kuras ML. 2006. Engineering Complex Systems. In *Complex Engineered Systems: Science Meets Technology*, Eds Braha D, Minai A A, Bar-Yam Y, pp 204-244, Springer, Cambridge, Massachusetts.

O'Neal MB, and Edwards WR. 1994. Complexity Measures for Rule-Based Programs. *IEEE Transactions on Knowledge and Data Engineering*, Vol 6, pp 669-680.

Ottino JM. 2004. Engineering Complex Systems. *Nature*, No 427, pp 399.

Pagels RH. 1989. *Dreams of Reason: The Computer and the Rise of the Science of Complexity*. Bantam Books, New York.

Pahl G, and Beitz W. 1996. *Engineering Design – A Systematic Approach*. Springer, New York, NY.

Papalambros PY. 2002. The Optimization Paradigm in Engineering Design: Promises and Challenges. *Computer-Aided Design*, Vol 34, pp 939-51.

Pimm S. 1984. The complexity and stability of ecosystems. *Nature*, Vol 307, pp 321-326.

Pimmler TU, and Eppinger SD.1994. *Integration Analysis of Product Decompositions.* Paper presented in ASME Design Theory and Methodology Conference, Minneapolis, MN, September.

Pramee IC, and Bonham CR. 2000. Towards the Support of Innovative Conceptual Design through Interactive Designer Evolutionary Computing Strategies. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol 14, pp 3–16.

Prasad B.1996. *Concurrent engineering fundamentals: integrated product and process organisation.* Prentice Hall, New Jersey.

Pugh S. 1991. *Total Design*. Addison Wesley, Wokingham, UK.

Quadrel RW, Woodbury RF, and Fenves SJ. 1993. Controlling Asynchronous Team Design Environments by Simulated Annealing. *Research in Engineering Design*, Vol 5, pp 88-104.

Reggiani A, and Nijkamp P. 1995. Competition and complexity in spatially connected networks. *System Dynamics Review*, Vol 11, pp 51-66.

Reidsema, C. 2001. *A Conceptual Blackboard Database Model For Design Process Planning In Concurrent Engineering.* PhD Thesis, University of Newcastle, Newcastle, Australia.

Reidsema C, and Szczerbicki E. 2002. Review of Intelligent Software Architectures for the Development of An Intelligent Decision Support System for Design Process Planning in Concurrent Engineering. *Cybernetics and Systems*, Vol 33, pp 629-658.

Ring J, and Madni A. 2005. *Key Challenges and Opportunities in 'System of Systems' Engineering.* In proceedings of IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, Hawaii, pp 973–978, October 10–12.

Roemer TA, and Ahmadi R. 2004. Concurrent Crashing and Overlapping in Product Development. *Operations Research*, Vol 52, No 4, pp 606-622.

Rosenman M, and Wang F. 1999. CADOM: A Component Agent-based Design-Oriented Model for Collaborative Design. *Research in Engineering Design*, Vol 11, pp 193–205.

Rusbult C. 2000. *Relationships between Design and Science (Part 1) Effective Combining of Creative and Critical Thinking.* American Scientific Affiliation, http://www.asa3.org/ASA/education/think/science-design.htm

Saad M, and Maher ML. 1996. Shared Understanding in Computer-Supported Collaborative Design. *Computer-Aided Design*, Vol 28, pp 183-192.

Sanders I. 2003. *What is Complexity?*. Washington Centre for Complexity and Public Policy, Washington. http://www.complexsys.org/pdf/what_is_complexity.pdf

Schruben L, and Ycesan E. 1993. *Complexity of simulation models: a graph theoretic approach.* In Proceedings of the 1993 Winter Simulation Conference,

Eds Evans GW, Mollashasemi M; Russell EC, et al, pp 641-649, IEEE, Piscataway, NJ.

Seeley D, and Ronald S. 1992. The Emergence of Connectivity and Fractal Time in the Evolution of Random Digraphs. *Complex Systems*, Vol 92, Australia National University.

Shalizi CR. 2001. *Causal Architecture, Complexity and Self-Organization in Time Series and Cellular Automata.* PhD Thesis, University of Wisconsin-Madison, US.

Simon HA. 1969. *Sciences of the artificial.* M.I.T. Press, Cambridge, MA.

Sinha R, Liang VC, Paredis CJJ, et al. 2001. Modeling and Simulation Methods for Design of Engineering Systems. *Computing and Information Science in Engineering*, Vol 1, pp 84-91.

Shen W, and Barthès JP. 1996. An Experimental Multi-Agent Environment for Engineering Design. *International Journal of Cooperative Information Systems*, Vol 5, pp 131-151.

Shen W, and Norrie DH .1998. *A Hybrid Agent-Oriented Infrastructure for Modeling Manufacturing Enterprises.* In proceedings of Eleventh Workshop on Knowledge Acquisition, Modeling and Management KAW'98, pp 117-128. Banff, Canada.

Shen W, Norrie DH, and Barthès J-P. 2001. *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing*. CRC Press.

Shi J, and Malik J. 2000. Normalized Cuts and Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 22, pp 888-905.

Smith RP, and Eppinger SD. 1997. Identifying Controlling Features of Design Iteration. *Management Science*, Vol 43, No 3, pp 276-93.

Soofi E. 1997. Information Theoretic Regression Methods. In *Advances in Econometrics - Applying Maximum Entropy to Econometric Problems*, Eds Fomby T, and Carter HR, Vol 12, Jai Press Inc., London.

Sosa ME, Eppinger SD, and Rowles CM. 2000. *Designing Modular and Integrative Systems*. Paper presented at International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Baltimore, Maryland, September.

Sosa R, and Gero J. 2004. Diffusion of Creative Design: Gate keeping Effects. *International Journal of Architectural Computing*, Vol 2, pp 518-531.

Sosa R, and Gero J. 2005. A Computational Study of Creativity in Design. *Artificial Intelligence for Engineering Design Analysis and Manufacturing*, Vol 19, pp 229-244.

Spielman DA, and Teng S.2007. Spectral Partitioning Works: Planar Graphs and Finite Element Meshes. *Linear Algebra and its Applications*, Vol 421, pp 284–305.

Sriram D, Stephanopoulos G, Logcher R, et al. 1989. Knowledge-Based System Applications in Engineering Design: Research at MIT. *AI Magazine*, Vol 10, No 3, pp 79-96.

Stacey R D. 1995. The Science of Complexity: An Alternative Perspective for Strategic Change Processes. *Strategic Management Journal*, Vol 16, pp 477-495.

Steel M. 1992. The Complexity of Reconstructing Trees from Qualitative Characters and Subtrees. *Journal of Classification*, Vol 9, pp 91-116.

Stepney S, Smith RE, Timmis J, et al. 2005. Conceptual frameworks for artificial immune systems. *Journal of Unconventional Computing*, Vol 1, No 3, pp 315–338.

Suh NP. 1988. *The Principles of Design*. Oxford University Press, Oxford, UK.

Suh NP. 1999. A Theory of Complexity, Periodicity and the Design Axioms. *Research in Engineering Design*, Vol 11, No 2, pp116–131.

Suh NP. 2001. *Axiomatic Design: Advances and Applications.* Oxford University Press, New York, NY.

Suh NP. 2005. *Complexity theory and applications.* Oxford University Press, New York.

Summers JD, Shah J. 2003. *Developing Measures of Complexity for Engineering Design.* Proceedings of ASME Design Engineering Technical Conferences DETC'03, September 2-6, Chicago, Illinois.

Sycara K. 1998. Multi Agent Systems. *AI Magazine*, Vol 19, pp 79-93.

Tanaka K, Higashiyama M, and Ohsuga S. 2000. *Problem Decomposition and Multi-agent System Creation for Distributed Problem Solving.* Proceedings of the 12th International Symposium on Foundations of Intelligent Systems, Springer-Verlag.

Temperly HNV. 1981. *Graph Theory and Applications.* Ellis Horwood, Chichister.

Terwiesch C, and Loch CH. 1999. Measuring the Effectiveness of Overlapping Development Activities. *Management Science*, Vol 45, No 4, pp 455-465.

Thomke S. 2001. Enlightened Experimentation: The New Imperative for Innovation. *Harvard Business Review*, Vol 79, No 2.

Timmis J. 2007. Artificial immune systems - today and tomorrow. *Natural Computing*, Vol 6, No 1, pp 1–18.

Timmis J, Andrews P, Owens N, et al (2008) an Interdisciplinary Perspective on Artificial Immune Systems. *Evolutionary Intelligence*, 1: 5-26

Tomiyama T, Umeda Y, Ishii M, et al (1995) Knowledge systematization for a knowledge intensive engineering framework. In *Knowledge Intensive CAD*, Eds Tomiyama T, Mantyla M, Finger S, pp 55-80, Chapman & Hall.

Turban E. 1995. *Decision support and expert systems: management support systems*. Prentice Hall, Englewood Cliffs, N.J.

Ulrich KT, and Eppinger SD. 2004. *Product Design and Development*. Third ed, Mc Graw-Hill/Irwin.

Unger DW. 2003. *New Product Process Design: Improving Development Response to Market, Technical, and Regulatory Risks.* PhD Thesis, MIT School of Management and Engineering, MA.

Verma D, and Meila M. 2003. *A Comparison of Spectral Clustering Algorithms*. Technical Report. University of Washington.

Watson AH, and McCabe TJ. 1996. *Structured Testing: A Testing Methodology Using the Cyclomatic Complexity Metric.* NIST (U.S. National Institute of Standards and Technology) Special Publication, pp 500-235, Washington.

Walker CC. 1971. Behaviour of a Class of complex systems: The effect of systems size on properties of terminal cycles. *Journal of Cybernetics*, Vol 1, pp 55-67.

Weber RG, and Condoor SS. 1988. *Conceptual Design Using a Synergistically Compatible Morphological Matrix.* In proceedings of Frontiers in Education Conference FIE '98, Vol 1, pp 171-176

White S, and Smyth P. 2005. *A spectral Clustering Approach to Finding Communities in Graphs.* In 5th SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2005.

Winograd S. 1963. Redundancy and Complexity of Logical Elements. Information and Control, Vol 5, pp 177-194.

Wolf TD, and Holvoet T. 2005. Towards a Methodology for Engineering Self-Organising Emergent Systems. In *Self-Organization and Autonomic Informatics*, Eds Czap H, Unland R, Branki C, et al, pp 18-34. Glasgow, UK

Wong A, and Sriram D. 1993. SHARED: An information model for cooperative NPD. *Research in Engineering Design*, Vol 5, pp 21-39.

Yeh CW, Cheng CK., and Lin T. 1992. *A probabilistic Multicommodity-Flow Solution to Circuit Clustering Problems.* Paper presented in IEEE Int. Conf. Computer-Aided Design ICCAD-92, Santa Clara, CA.

Zadeh LA. 1973. Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. On Systems, Man and Cybernetics*, No 3, pp 28-44.

Zhang C. 1992. Cooperation under uncertainty in distributed expert systems. *Artificial Intelligence*, Vol 56, pp 21-69.

Zeigarnik AV, and Temkin ON. 1996. A graph-theoretic model of complex reaction mechanisms: a new complexity index for reaction mechanisms. *Kinetics and Catalysis*, Vol 37, pp 372-385.

Zdrahal Z, and Motta E. 1996. *Case-Based Problem Solving Methods for Parametric Design Tasks.* In proceedings of the Third European Workshop EWCBR-96 Lausanne, November 14–16, Switzerland, pp 473-486.

A Appendix : The Nondisclosures and Matlab™ Codes

A.1 Graph Theoretic Complexity Measure

The graph theoretic measure of complexity of the Ontospace™ software (and this thesis) is the Spectral Norm of the adjacency matrix of the graph. In general, the Spectral norm of a matrix *A* is also known as the spectral radios (or second norm) of the matrix and is the largest singular value of *A*. For real matrices the spectral norm is:

$$\|A\|_2 = \sqrt{\lambda_{max}(A^T \times A)}$$

Where A^T is the transpose of matrix A. In general, a norm is an abstraction of the concept of length. In robust control theory[22] the spectral norm of a controller transfer function is known as H_∞ or Hankel norm, and is used to minimize the closed loop impact of a perturbation of outputs to input perturbation. There is an analogy between robust controller design and robust engineering design. In fact a controller transfer function in control theory is the equivalent of a DSM in design theory. Robust control design has become a new paradigm in modern control design already many years ago. However the introduction of this technique into engineering design which is also accompanied by its mix with the science of emergence is relatively new.

The notion of emergence has very close relationship with this measure. The emergence of connectivity in random graphs (Erdős–Rényi graphs) is usually analysed in terms of the size of the largest connected subgraph (LGS) (Boschetti et al 2005). A subgraph of a graph *G* is a graph whose vertex set is a subset of that of *G*, and whose adjacency relation is a subset of that of *G* restricted to this subset.

[22] Robust control theory is concerned with how control systems react to erroneous or failed inputs or stressful environmental conditions.

A subgraph is connected if only if it cannot be any further partitioned into two or more independent components (or subgraphs with no edge in between them). Spectral norm is obviously a property of the LGS since if we consider matrix *A* that is collection of sub-matrices on its diagonal and zero elsewhere then the spectral norm of A would be the maximum of the spectral norms of the sub-matrices:

$$\|A\| = \left\| \begin{bmatrix} A_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & A_n \end{bmatrix} \right\|_2 = Max_{i=1..n}(\|A_i\|_2)$$

Note that A_is can be the adjacency matrices of the sub-graphs of a graph (with adjacency matrix A) that are not connected to each other. We used this property of this measure in chapter 5 to set the lower bound for the *real complexity* of decompositions. Figure A.1 shows that on average the measure increases with three components of complexity (order, size or coupling, and cycles). In this figure each point represents a randomly generated graph (see Section A.3 for the related Matlab™ code).

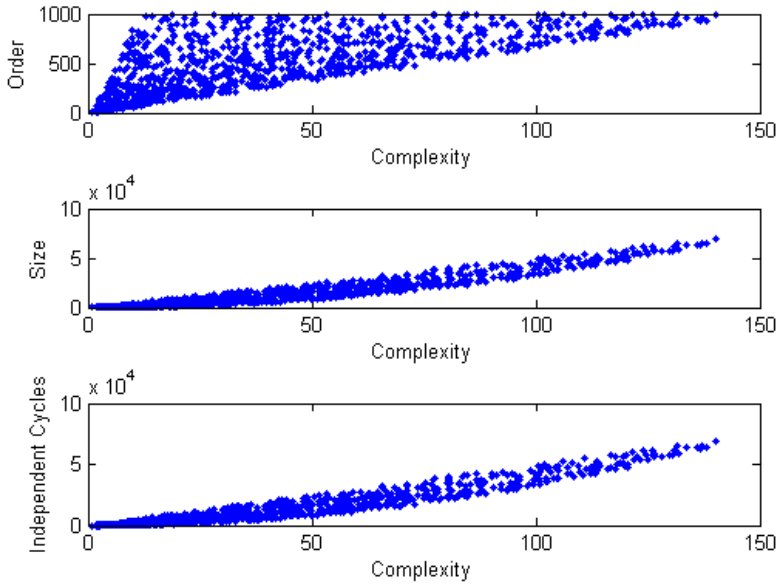


Figure A.1 The spectral norm as a complexity measure on average increases with size, order and cycles number.

A.2 Lower and Upper Complexity Bounds

The lower and upper complexity bounds are measured from a dataset that describes a system and by applying the complexity measure to respectively a reduced adjacency matrix and an augmented one. The reduced adjacency matrix contains only the *important* edge weights and is zero for the weak links. This way the reduced adjacency matrix expresses the *essence* of connectivity in the system and the *irreducible core* of the system. Ontospace^{TM} computes the information exchange (by estimating the mutual entropy) and the entropy of all the scatter plots for the pairs of systems variables (by treating the scatter plots as images). The scatter plots that contain significant information exchange (the important links in the system's graph) are decided by drawing a boundary in the plot of Entropy versus Information Exchange of all scatter plots of pairs of system's variables (Figure A.2). This boundary separates the scatter plots that contain useful information (shown by red dots) from the remainder (blue dots).

Generation of the augmented adjacency matrix involves Monte Carlo sample generation to fill up each individual scatter plot with more samples points (based on the estimated mutual probability distributions from the original data set). Naturally, every time a new sample is generated the entropy in the scatter plot increases. The augmented adjacency matrix is formed by measuring the information exchanges when all scatter plots have the maximum entropy as shown in the Figure A.3 (i.e. all scatter plots would be saturated with the maximum entropy).

In chapter 6 we mentioned the notions of real minimum and maximum complexities for the hierarchies. Real minimum and maximum complexities are obtained by decomposing the reduced and augmented adjacency matrices of the system, then measuring the real complexity for these decompositions.

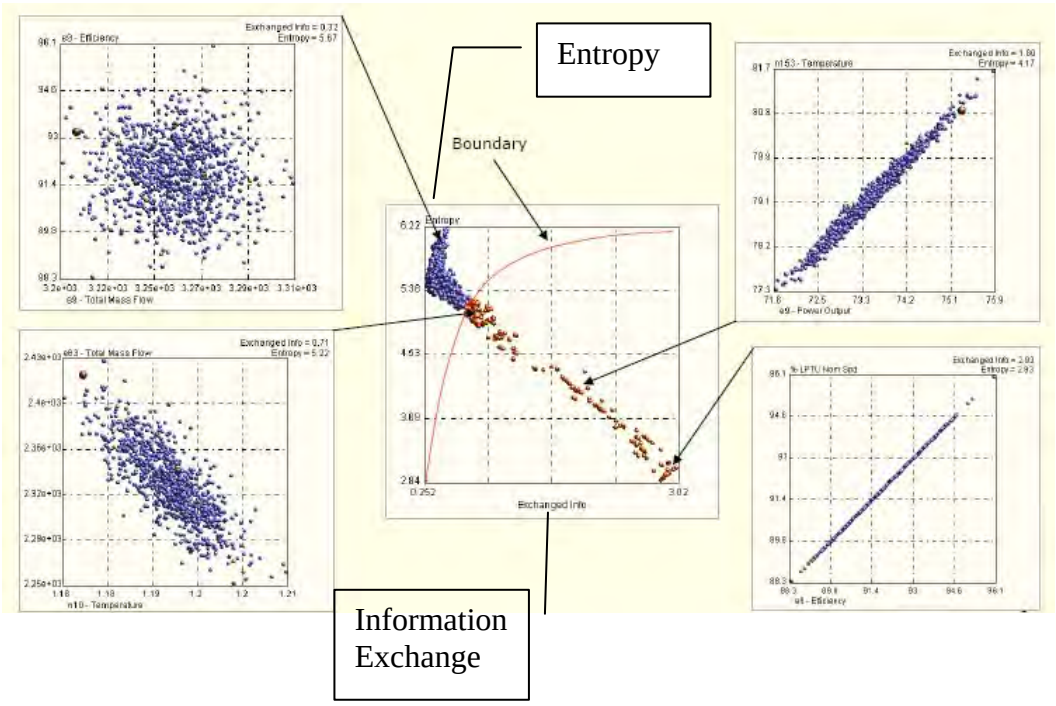


Figure A.2 The scatter plots that contain useful information are separated from those that do not.

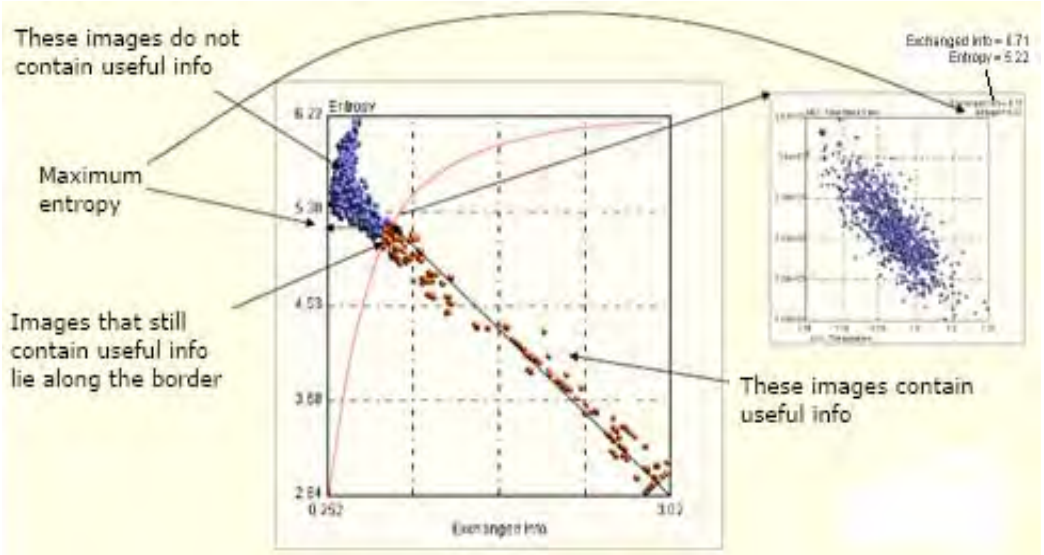


Figure A.3 The maximum entropy is the entropy of the image that lies on the boundary.

A.3 MATLAB™ Codes

A.3.1 Codes related to Chapter 3

Function: figure3_6

This function generates the Figure 3.6 and the mutual information exchange presented in the corresponding section.

```
function figure3_6
x = linspace(-1,1,200);
y  = xcircle(x);
x = [x x];
tt(:,1) = x';
tt(:,2) = y';
plot(x,y,'.')
mutual information = xentropy( tt );
```

Function: xcircle

This function generates a half circle.

Output:

- *y*: a vector containing the y coordinates of a circle.

Input

- *x*: a vector containing the x coordinates of the circle.

```
function y = xcircle (x)
y1 = (1 - x.^2 ) .^ 0.5;
y2 = -((1 - x.^2 ) .^ 0.5);
y = [y1 y2];
end
```

Function: xentropy

This function estimates the joint probability distribution of two random variables and their mutual information.

Output:

- *ent*: the mutual information

Input:

- *tt*: a vector with two columns. Each row of the vector is a data point of a two dimensional data set.

```
function ent = xentropy( tt )
        h = hist3(tt);
        N = sum(sum(h));
        w = h / N;
        q = find(w(:,:) ~= 0);
        ent =  - sum( sum( w(q) .* log ( w(q) ) ) ,  2)
end
```

A.3.2 Codes Related to Chapter 5

A number of Matlab™ functions have written and used to produce the presented figures of Chapter 5. These functions are included next.

Function: dcm

This function measures the real complexity of decompositions.

Outputs:

- *com*: real complexity of the decomposition
- *Csum*: sum complexity of the parts ort subsystems
- *rcut*: cut ratio of Cheng and Hu (1989)
- *mcut*: min-max cut of Ding et al (2001)

- *ncut*: normalized cut of Shi and Malik (1991)

Inputs:

- *a*: adjacency matrix of the self, can be weighted or un-weighted.
- *m*: partitions (row) vector. Each of the elements of this vector are the order of one of the partitions, in a way that, the size of the vector is the number of the partitions, and the sum of the elements of the vector are the order of the adjacency matrix of the self graph.
- *pn*: the permutation (row) vector of the self adjacency matrix. Its size is the same as the size of self adjacency matrix and contains the permutation indices. For example if the matrix *A* is a 3×3 self adjacency matrix then the permutation vector can be one of the following six vectors:

  [3 2 1], [2 3 1], [3 1 2], [2 1 3], [1 3 2] and [1 2 3]

  The permutation of *A* with the last vector would be equal to *A*.

```
function [  com, Csum, rcut , mcut , ncut   ] = dcm(a,m,pn)
a_n = a(pn,pn);
nsus = length(m);
nim = cumsum(m);
for ii = 1:nsus
            Sus(ii)= {a_n( nim(ii) - m(ii) + 1 : ...
                nim(ii), nim(ii)- m(ii)+ 1 : nim(ii))};
                      for jj = 1:ii-1
                   cut(jj,ii)={a_n( nim(jj) - m(jj) + 1 : nim(jj),...
                       nim(ii) - m(ii) + 1 : nim(ii))};
                   cut(ii,jj)={a_n( nim(ii) - m(ii) + 1 : nim(ii),...
                       nim(jj) - m(jj) + 1 : nim(jj))};
             end
             cut(ii,ii) = {[]};
end
            Cum  = cellfun(@norm, Sus);
            sums = sum(Cum);
            smin = min(Cum);
            subcom = cellfun(@(x) sum(sum(x)), Sus);
            Csum = sum(Cum);
            Cum = diag(Cum);
            cutsum = cellfun(@(x) sum(sum(x)), cut);
            cum = Cum + cutsum;
            com = norm(cum);
            rcut = sum(sum(cutsum)./m)/nsus/(nsus-1);
            mcut = sum (sum (cutsum) ./  subcom) / nsus/(nsus -1);
            ncut = sum (sum (cutsum)  ./ (sum (cutsum) + subcom))/
            nsus/... (nsus -1);
```

```
end
```

Function: Overlapdcm

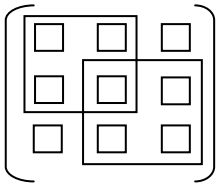
This function calculates the real complexity of the overlap decompositions.

Outputs:

- *com*: real complexity of the overlap decomposition

Inputs:

- *a*: adjacency matrix of the self graph
- *m*: the partitions (row) vector. This includes partitions size including the overlap blocks. For example if the matrix “a” is 3×3 and it is decomposed in the following form:



  Then the partitions matrix is written as [1 1 1].
- *l*: the vector containing the indices of the laps in the partition vector. For the above example, l would be [2].
- *pn*: the permutation row vector containing the permutation indices of adjacency matrix.

```
function [com] = overlapdcm(a,m,l,pn)

mm = size(m,2);
ll = size(l,2);
I = zeros(mm+ll,mm);
j = 0;
for i = 1:mm
    j = j+1;
    I(j,i)=1;
    if sum(i == l)== 0
    else
```

```
        j = i + find(i==l);
        I(j,i) = 1 ;
    end

a_n = a(pn,pn);
nsus = length(m);
nim = cumsum(m);
mm = m;
mm(l-1) = mm(l) + mm(l-1);
mm(l)=[];
mim = cumsum(mm);
mm = m;
mm(l-1) = mm(l) + mm(l-1);
mm(l+1) = mm(l) + mm(l+1);
mm(l)=[];
msus = length (mm);
for i = 1:msus
    b = [mim(i)-mm(i)+1 : mim(i)];
    Mb(i) = {a_n(b,b)};
end
parts = cellfun(@norm,Mb);
for ii = 1:nsus
            Sus(ii)= {a_n( nim(ii) - m(ii) + 1 : ...
                nim(ii), nim(ii)- m(ii)+ 1 : nim(ii))};
                          for jj = 1:ii-1
                   cut(jj,ii)={a_n( nim(jj) - m(jj) + 1 : nim(jj),...
                       nim(ii) - m(ii) + 1 : nim(ii))};
                   cut(ii,jj)={a_n( nim(ii) - m(ii) + 1 : nim(ii),...
                       nim(jj) - m(jj) + 1 : nim(jj))};
               end
                     cut(ii,ii) = {[]};
                                              end
            Cum  = cellfun(@norm, Sus);
            Cum = diag(Cum);
cutsum = cellfun(@(x) sum(sum(x)), cut);
Cum = Cum + cutsum;
cum = I*Cum*I';
com = norm(cum);

end
```

Function: adj

This function generates an adjacency matrix of an unweighted graph.

Output:

- *a*: the unweighted adjacency matrix

Inputs:

- *n*: the order of the graph or the size of the adjacency matrix.
- *p*: the probability that an edge exists between two vertices.

```
function a = adj(n,p)
if p>1 | p<0, error('p must be between zero and one')
else
a = rand(n)< p;
a = triu(a,1);
a = a+a';
end
```

Function: adjweid

This function changes an unweighted adjacency matrix to a weighted one.

Output:

- *a*: the weighted adjacency matrix.

Inputs:

- *a*: the unweighted adjacency matrix.
- *p1*: depending on the value of *l*, can be the mean or the minimum of the edge weights.
- *p2:* depending on the value of *l*, can be the standard deviation or the maximum of the edge weights.
- *l*: can be either 1 or 2. If *l=1* then a normal distribution is used for the distribution of the weights. If *l=2* then a uniform distribution is used.

```
function a = adjweid(a,p1,p2,l)
i = find(a);
j = nnz(a);
switch l
    case 1%p1 = mean,p2 = std
        a(i) = sqrt(p2) * randn(1,j) + p1 ;
        a(i) = a(i) .* ( a(i) > 0 );
            case 2%p1 = min,p2 = max
```

```
        a(i) = (p2 - p1) * rand(1,j) + p1 ;

    otherwise
        error('l must be zero or one')
end
end
```

Function: fdlr

This function produces the permutation vector based on the sorted indices of the eigenvectors of the adjacency and Laplacian matrices.

Outputs:

- *pn*: the permutation vector

Inputs:

- *a*: adjacency matrix (can be weighted or unweighted)
- *i*: can be either 0 or 1

```
function [pn] = fdlr(a,i)
% m = zero -> adjacency
% m = 1 -> laplacain
n = length(a);
 L = diag(sum(a,2)) - a ;
 [v1 v2] = eig(L);
[u1 u2] = eig(a);
if i == 0
[x1 pn] = sort(u1(:,n));
elseif i==1
[x1 pn] = sort(v1(:,n));
end
```

Function: kpart

This function produces a row vector that contains the size of the partitions. The size of this vector is the number of partitions. The partition sizes are balanced since they are drawn from a normal distribution. The minimum and maximum size of the partitions can also be stated.

Outputs:

- *m*: partitions raw vector

Inputs:

- *n*: order of the original graph that is to be partitioned.
- *k*: the number of partitions
- *mu*: the mean size of the partitions.
- *sigma*: the standard devisation of the size of partitions from the mean.
- *smin*: the minimum size of the partitions.
- *smax*: the maximum size of the partitions.

```
function m = kpart(n,k,sigma, mu, smin,smax)
m = zeros(1,k);
while nnz(m > smax) || nnz(m < smin)
m = mu * randn(1,k) + sigma;
  m = m * n / sum(m);
m = ceil(m);
[jj,j] = max(m);
m(j) = m(j)- (sum(m)-n);
p = randperm(k);
m = m(p);
end
```