UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Enhancing the conceptual design phase of complex engineering systems with an integrated methodology and support tools

Ivo Miguel Lopes Ferreira

Supervisor: Doctor Paulo Jorge Soares Gil

Thesis approved in public session to obtain the PhD Degree in

Leaders for Technical Industries

Jury final classification: Pass with Merit

Jury

Chairperson: Chairman of the IST Scientific Board
Members of the Committee:
Doctor Olivier Ladislas de Weck
Doctor Elsa Maria Pires Henriques
Doctor Rui Manuel de Sá Pereira de Lima
Doctor António Manuel Relógio Ribeiro
Doctor Paulo Jorge Soares Gil

2012

UNIVERSIDADE TÉCNICA DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

Enhancing the conceptual design phase of complex engineering systems with an integrated methodology and support tools

Ivo Miguel Lopes Ferreira

Supervisor: Doctor Paulo Jorge Soares Gil

Thesis approved in public session to obtain the PhD Degree in

Leaders for Technical Industries

Jury final classification: Pass with Merit

Jury

Chairperson: Chairman of the IST Scientific Board
Members of the Committee:
Doctor Olivier Ladislas de Weck, Professor Associado, Massachusetts Institute of Technology
Doctor Elsa Maria Pires Henriques, Professora Associada, Instituto Superior Técnico, Universidade Técnica de Lisboa
Doctor Rui Manuel de Sá Pereira de Lima, Professor Auxiliar, Escola de Engenharia, Universidade do Minho
Doctor António Manuel Relógio Ribeiro, Professor Auxiliar, Instituto Superior Técnico, Universidade Técnica de Lisboa
Doctor Paulo Jorge Soares Gil, Professor Auxiliar, Instituto Superior Técnico, Universidade Técnica de Lisboa

Funding Institutions

Fundação para a Ciência e a Tecnologia

2012

I. ABSTRACT

The impact of conceptual design is well known in the context of complex engineering systems, changes made during later design phases can severely impact the performance of the design process and the quality of the final outcome. This thesis proposes an integrated methodology to deal with the shortcomings of current collaborative design environments, identified from both literature and survey findings.

The proposed integrated methodology is based on the premise that the design properties can emerge after a proper identification of the system-level objectives, and consequent assessment of their impact at the subsystem level. It is an iterative process with four clear stages that monitors the evolution of objectives, requirements, system states, and subsystem design models.

This thesis also describes the development of two tools to support the integrated methodology: a predictive decision support tool, and an integrated concurrent environment depicting the interactions between subsystems with the help of a dynamic Design Structure Matrix.

Through a set of case studies, this work provides a quantitative assessment of the gain obtained by using predictive decision support tools, and suggests a qualitative improvement in the global design process by using an integrated design tool during the conceptual design phase.

Keywords: Complex engineering systems, Systems engineering, Conceptual design, Concurrent Engineering, Collaborative design environments, Neural networks, Decision support

[*This page intentionally left blank*]

II. RESUMO

O impacto do design preliminar no contexto de sistemas de engenharia complexos é bem conhecido, alterações feitas após esta fase podem criar efeitos muito negativos tanto no desempenho do processo de design, como na qualidade da solução produzida. Esta tese compila uma série de ideias e directrizes numa metodologia para resolver alguns dos problemas das actuais ferramentas de design colaborativo, identificadas tanto na literatura como através de inquéritos.

A metodologia proposta parte da premissa de que as principais propriedades do design podem emergir através de uma identificação correta dos objectivos ao nível do sistema global, e de um mapeamento do impacto destes ao nível de cada subsistema. Trata-se de um processo iterativo, com quatro fases distintas, para monitorar a evolução dos objectivos, requisitos, estados do sistema e modelos de subsistemas.

Esta tese também descreve o desenvolvimento de duas ferramentas para implementar algumas das ideias introduzidas na metodologia: uma ferramenta de apoio à decisão baseada na previsão obtida por modelos de inteligência artificial multi-agente e, uma outra para suportar a integração dos diferentes subsistemas através de uma monitorização das interações entre subsistemas com a ajuda de uma Design Structure Matrix dinâmica.

Os casos de estudo desenvolvidos permitiram uma análise quantitativa do ganho no apoio à decisão, e sugerem um ganho qualitativo no processo de design global usando o ambiente concorrente proposto.

Palavras-chave: Sistemas complexos de engenharia, Engenharia de sistemas, Design preliminar, Engenharia concorrente, Ambientes de design colaborativos, Redes neuronais, Suporte à decisão

[*This page intentionally left blank*]

III. ACKNOWLEDGMENTS

It is ironic to realize the self-fulfilling prophecy of this work. With the ultimate goal of dealing with the complexity of engineering systems, this research became, by itself, a complex system. This thesis describes the technical complexity of the final outcome of this four-year process, but it hides some of the collaborative actions that led to the research path chosen.

My name might be the one that appears on the by-line of this thesis, but it is the product of many contributions from a large number of individuals that are part of three different organizations: Instituto Superior Técnico - Lisbon - Portugal (IST), Massachusetts Institute of Technology - Cambridge - MA - USA (MIT), and the European Space Research and Technology Centre of the European Space Agency - Noordwijk - The Netherlands (ESTEC).

First, I would like to thank my advisor, Prof. Paulo Gil from IST, for his continuous supervision and support. I really appreciated the advices and contributions to make this research experience stimulating and motivating. I would also like to thank the other members of the committee: Prof. Olivier de Weck from MIT and Prof. Rui Lima from Universidade do Minho, for their thoughtful ideas and comments.

From IST, I would like to thank Prof. Manuel Freitas, Prof. Elsa Henriques, Prof. Mihail Fontul, Prof. Relógio Ribeiro, Prof. Paulo Peças and Prof. Filipe Cunha for their inputs with the right amount of encouragement and criticism during the discussion sessions of this research work; your ideas really helped me when I was starting to veer of course. I would also like to show my gratitude to Prof. António Rito Silva, for lending me some of his expertise in model-based software architectures when I started developing INCREMENT. I would also like to show my appreciation to the undergraduate students of the aerospace engineering degree who took part in the case studies to test the predictive decision support tool during the fall semester of 2010/2011.

Still from IST, I would like to sincerely acknowledge all my colleagues with whom I shared research ideas, excitements and common frustrations: Marco Leite, Pedro Marques, Carla Pepe, Eduardo Santos, Bruno Soares, Raquel Folgado, Inês Ribeiro, João Fernandes, Jean-Loup Loyer, Nuno Oliveira, Paulo Esteves, Yorgos Koronis and Ioannis Malliaros. You

have taught me that there is nothing better than a coffee and a laugh to put ideas straight again.

The period spent at the MIT was undoubtedly one of the most fruitful ones on this work. I would like to thank Prof. Olivier de Weck for receiving me in his research group. It was a truly enlightening experience to interact with such a brilliant group of individuals. I am especially grateful to Paul Grogan for all your help and support with SpaceNet, the departure point to some of the tools I developed later. I am also thankful for the assistance of the MIT Portugal Program staff during this period: Prof. Robin Lemp, Prof. Jeremy Gregory and Gerri Powers.

The internship period spent at the ESTEC brought a real context to this work. I am also indebted to many supportive and generous people at ESA who shared with me their vision on the problems and possible enhancement of collaborative design tools through interviews or surveys. I am especially thankful to Massimo Bandecchi and Tiago Soares for giving access to some of the details of the CDF environment.

This PhD was only possible due to the MIT-Portugal Program. I would like to thank all of the people that, one way or the other, contributed to the existence of this program: Prof. Manuel Heitor, Prof. Paulo Ferrão, Prof. António Torres Marques, Prof. António Cunha, Prof. Manuel Freitas, amongst many others. I also very grateful for the scholarship of "Fundação para a Ciência e Tecnologia" of the Portuguese Government which provided me with all the necessary funding.

Finally, I would like to acknowledge the pillars of my construction: my family. I would especially like to thank my mother (Lucília Lopes), father (João Ferreira), and sister (Joana Ferreira) for always being there supporting me on every decision. This list would not be complete without mentioning the person who shared all the moments of this thesis with me; Ivy, thank you for everything.

IV. TABLE OF CONTENTS

[*This page intentionally left blank*]

V. LIST OF FIGURES

[*This page intentionally left blank*]

VI. LIST OF TABLES

VII. LIST OF ACRONYMS

ANN – Artificial Neural Networks

CDF – Concurrent Design Facility

CE – Concurrent Engineering

CMM – Capability Maturity Model

CSCD – Computer Supported Collaborative Design

DeMAID – Design Manager's Aid for Intelligent Decomposition

DIKW – Data, Information, Knowledge and Wisdom

DMM – Domain Mapping Matrices

DOE – Design Of Experiments

DSM – Design Structure Matrix

ERM – Entity Relationship Model

ESA – European Space Agency

ESTEC – European Space Research and Technology Centre

GEO – Geosynchronous Earth Orbit

INCREMENT – INtegrated Concurrent Real-time EnvironMENT

IT – Information Technology

KM – Knowledge Management

LEO – Low Earth Orbit

MAUA – Multi Attribute Utility Analysis

MBSE – Model Based System Engineering

MCDA – Multi-Criteria Decision Analysis

MDO – Multidisciplinary Design Optimization

MODA – Multi-Objective Decision Analysis

NASA – National Aeronautics and Space Administration

OCDS – Open Concurrent Design Server

OPD – Object-Process Diagram

OPL – Object-Process Language

OPM – Object-Process Methodology

PDM – Product Data Management

PLM – Product Lifecycle Management

SE – Systems Engineering

SysML – Systems Modeling Language

UML – Unified Modeling Language

VDSM – Visual Design Structure Matrix

VM – Visualization Methods

VRML – Virtual Reality Modeling Language

1. INTRODUCTION

Agostinho da Silva, one of the most prominent Portuguese philosophers of the 20th century, argued that this century will be the century where humans become «poetas à solta» (wandering poets), where technology will gradually deal with all repetitive and incremental tasks and humans will be destined to knowledge-intensive, disruptive, and innovative advances. What we have been experiencing in the last few decades suggests that technology is playing its role in this vision, we are on the verge of reaching the so called singularity point (Kurzweil 2005), at which the exponential growth of our technological capabilities will empower greater than human intelligence. We should commit ourselves to play our role and focus all these powerful resources to amplify creativity and focus on ever growing real world complex applications.

The design process is a way through which we can affirm ourselves as «poetas à solta», turning innovative ideas into new systems by bridging the gap between the dreams of tomorrow and the reality of today. Every contribution to help increase the quality of this process can have a severe influence on the way we will answer tomorrow's questions. Within the design process, it is well known that its early stages have a considerable impact on the outcome. Design changes tend to be extremely difficult and expensive at later stages of the product design where a considerable amount of the project budget is already committed. The initial phase of product design is typically responsible for only 5% of the cost of a product. However, during this phase 75% or more of all manufacturing costs are committed, and about 80% of the system's quality performance is determined (Dowlatshahi 1999; Huthwaite 1988).

This thesis builds on the work of others who have delved into the design process issues, and proposes collaborative tools to apply during the early design phase. This research tackles the issues undermining performance or causing excessive cost during the design process.

1.1. Objective and motivation

The main purpose of this work is to introduce a new methodology and support tools for improving the early design phases of complex engineering systems. By analyzing the current shortcomings of collaborative design environments both from literature (Chapter

2) and survey findings (Chapter 3), this work introduces a potential way to approach the following fundamental needs:

- definition of common representations for every type of complex engineering systems,
- establishment of a shared integrated environment encompassing the model definition and the design process management tasks,
- incorporation of lessons learned from earlier design processes,
- transparent representation of historic evolution of the design process,
- traceability and support of design decisions.

These five higher level needs ultimately led to the proposal of an integrated conceptual design methodology detailed in Chapter 4, and two enabling tools:

- a predictive decision support tool using agent-based artificial neural network models to support early design decisions (Chapter 5), and,
- a higher level tool to support the overall design process named INtegrated Concurrent Real-time EnvironMENT (INCREMENT) (Chapter 5).

1.2. Context and fundamental concepts

The objectives and motivation of this work introduce a few global ideas that need to be dissected, along with the lower level concepts and inherited assumptions that guided the research process which resulted in this thesis.

The first key concept relates to the definition of a *system*. This has philosophical implications mainly linked to the characterization of the boundary conditions but, for the purpose of this work, a system "is an instance of a set of elements having relationships with one another sufficiently cohesive to distinguish the system from its environment thereby giving the system an identity" (Magee & de Weck 2004). A system is simply a set of parts that share commonalities and whose behavior is better understood by looking at these parts together than one by one.

The second concept deals with the meaning of *complexity* which is one of the few agreed-upon characterizations of how the world is today. Complexity is usually applied in the sense of intricacy or unpredictable issues that cannot be easily captured with reductionist approaches. This notion of complexity is not new, but its meaning certainly changed with our increasing ability to deal with it, largely due to the rapid growth of computing and

Information Technology (IT). What a few years ago was perceived as an extraordinarily complex system, today is fairly well understood and can be part of a much more complex system (system of systems). It is crucial to have an encompassing definition of what is a complex system but, more importantly, understand that this concept can only be fully understood in the relative sense and is dependent on the context and time at which it was defined.

A *complex system* can be defined as a system with numerous components, interactions or interdependencies that are difficult to describe, understand, predict, manage, design, and/or change (Boccaletti et al. 2006). It is a system that either by design, function or both is difficult to understand and verify. These systems exhibit an emergent behavior in the sense that some of its characteristics can only be understood by looking at it in a global perspective. The behavior of a complex system is more than the sum of the behavior exhibited by its parts. Weber proposes three dimensions of complexity in a product or system (C. Weber 2005): numerical, relational, and variational, and two more related to the process: disciplinary and organizational.

There have been some attempts to provide, mainly in the IT field, a standard metric for this concept by defining the complexity of a system as the time it takes for the fastest program running on a universal computer to compute the solution to the problem (Gill 1977). The use of a universal computer answers the need for a relative metric but makes the assumption that the system in question can be totally represented through a set of algorithms, i.e. its behavior can be explained by a set of rules commonly described as a Touring machine. This is often not the case for large systems exhibiting adaptive parts or involving the so called soft-skills that cannot be accurately modeled through common models (Lindemann et al. 2009).

The present work approaches complexity in the context of *engineering systems* by recognizing three relevant dimensions (Figure 1):

- *size and intricacy,* in terms of the number of items/subsystems, dependencies and required interactions amongst them,
- *technical difficulty*, since the disciplines involved require a level of understanding and background that cannot be easily perceived from the perspective of a non-specialist. Typically, a single individual cannot understand the whole system, being only a specialist on a specific domain, and,

- *sociocultural issues*, because they can be designed and managed by different organizations, with different underlying objectives (for instance research institutes vs. companies), involving people with different social and cultural backgrounds that cannot be neglected.

Modeling complex engineering systems requires a considerable effort described in a combination of these dimensions. Some systems can be complex simply due to an extremely high level of technical complexity, while others may be technically less exigent but still be complex as a result of their size, and/or sociocultural issues of the different stakeholders on the design process.

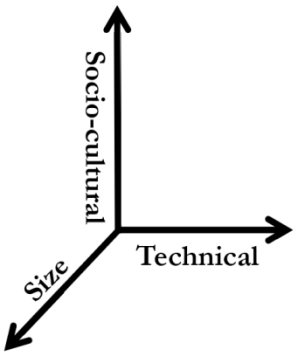


Figure 1: The dimensions of complex engineering systems

The main object of this work is the *design process* which can be defined as the actions of conception, invention, visualization, calculation, marshaling, refinement, and specifying of details which determines the form of a system (French 1985). The design process corresponds to the beginning of a system's lifecycle during which the main attributes are defined through a decision-making iterative process to match the early needs with the expectations. This iteration should encourage a clearer understanding of the underlying drivers of every decision.

This work focuses on the early design stage, which is commonly referred to as the *conceptual design phase* (Magee & de Weck 2004; D. Whitney et al. 1999) also known as preliminary or architectural design phase. It is during this phase that the first high level decisions are made, exploring synergies, user needs, and other elements, synthesizing what will eventually drive the final design. Even though the impact of this phase varies from system to system, being slightly lower for systems describing manufacturing processes than for products (Ulrich & Pearson 1993), it is at this point that the first architectural choices are made which results in a particularly significant influence on the final design cost and performance

(Dowlatshahi 1999; Huthwaite 1988). A large percentage of the cost is already committed while only a small portion is actually spent (Figure 2).

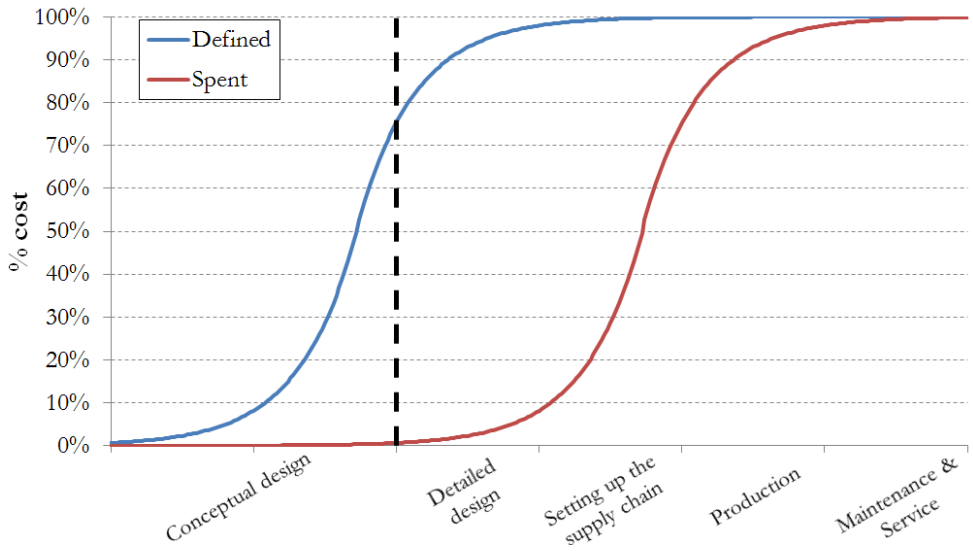


Figure 2: Impact of the conceptual design phase (adapted from (Nevins 1989; Hartley 1992))

The breadth and depth of the conceptual design phase have been the subject of significant literature and government documents (ECSS-TM-E-10-25A 2010; Kusiak 1993; NASA 2007; D. Whitney et al. 1999). This discussion is mainly tied with the definition of *requirements specification*. Some argue that there should be certain formal requirements before the conceptual design phase to guide the design process and increase its performance (Dowlatshahi 1994); others support that the first requirements should only arise once there already is a conceptual solution to meet a particular need and, therefore, should only arise after this phase is completed in order to prevent any hints from the pre-design conditions (Larson 1997).

This work considers a compromise between these two views and considers that the conceptual design phase departs from a preliminary definition of the design objective and the overall high level requirements, and results in a first list of possible conceptual architectures and more detailed requirements that later need to be approached, during the detailed design phase. In this context, the objective of the conceptual design phase is the development of a baseline concept that implements the initial requirements, assuming that the first high level requirements have been developed, assessed, and formalized.

The *performance metric* introduced in the research motivation relates to both the quality of the system itself (result of the design process) and the design process. This work approaches the former metric by looking at objective parameters such as the weight or cost of the

system. It considers that the latter metric is not only related to the time spent on the design process but also to a concept defined as insight.

Increasing the system's *insight* during the design process is a clear objective of this work. There is a certain resistance to accept a formal definition of insight believing that it would be either too general or too restrictive. North suggests that insight has the following essential characteristics (North 2006):

- complexity, involving large amounts of data in a synergistic way;
- depth, insight builds over time, accumulating and building on itself;
- subjectivity, insight is qualitative and requires a certain level of abstraction and creativity;
- relevancy, connecting data to knowledge and giving it meaning.

When dealing with complex engineering systems, the conceptual design phase is usually accomplished in a team, through a *collaborative design* environment. Since the conceptual design typically involves individually modeling the different lower level parts to iteratively converge towards possible design solutions, the degree of collaboration plays a crucial role in the ability to clearly and efficiently support and record the detailed needs of every stakeholder in the process, as well as the major design decisions and their rationale.

A collaborative design environment consists of two principal modes of communication: synchronous and asynchronous. The *synchronous* mode refers to the real-time communication between designers (same timeframe); the *asynchronous* mode allows different timeframes. Both have advantages and drawbacks: the synchronous mode is usually faster but more expensive since it involves more complex logistics to bring everyone together at the same time; the asynchronous mode is more convenient, but is slower and impersonal. The success of a collaborative design environment depends on a successful compromise between both these modes.

The advent and subsequent exponential progress of IT led to the establishment of a whole new field designated as *Computer Supported Collaborative Design* (CSCD), presented in section 2.5. CSCD is the process of designing a product through computer-enabled collaboration among multidisciplinary individuals associated with all phases of design (Sprow 1992). An important objective of CSCD is to support collaborative design, information and communication technologies that can be used to increase the quality of both human-human and human-machine interactions (Shen et al. 2008).

This work focuses on the application of CSCD to the conceptual design phase in a *Concurrent Engineering* (CE) environment. The concept of CE was initially proposed in the late 1980s as a potential means to minimize product development time. It was defined as a "systematic approach to the integrated, concurrent design of products and their related processes, including manufacture and support" (Turino 1992), the background of this concept is detailed in Chapter 2.

The basic premise for CE revolves around two concepts. The first is the idea that there is a gain in the system's final performance by considering, as many aspects as possible, as early as possible, into the design phase. Considering, right from the beginning, issues such as functionality, assembly, testability, environmental impact, or final disposal, can yield a large profit in terms of quality, lower cost, or lower resources spent. The second is that the preceding design activities should all be occurring at the same time, or concurrently. The overall goal being that the concurrent nature of these processes significantly increases productivity and product quality, aspects that are obviously important in today's fast-paced market (Kusiak 1993).

This design philosophy allows the early identification of errors and redesigns when the project is still in its early stage. By locating and fixing these issues early, the design team can avoid what often becomes expensive errors as the design process evolves to more complicated computational models and eventually into the final solution. Examples from companies using CE techniques show significant increases in overall quality, 30-40% reduction in project times and costs, and 60-80% reductions in design changes after release (Stark 2011).

Concurrent Engineering is based on three key elements: a process, a multidisciplinary team, and a software infrastructure (Bandecchi et al. 2000). The *process* should describe the logical sequence of tasks that need to be performed to accomplish the conceptual design, defining «what» needs to be done without specifying the «how». The different approaches to this process are detailed in section 2.1. The *multidisciplinary team* refers to the establishment of teamwork culture, with effective communication, between the different experts involved in the design, the manufacturing teams, and even the customer (Balamuralikrishna et al. 2000). The *software infrastructure* is the collaborative software (also referred to as groupware, work group support systems or simply group support systems) designed to help people involved in common tasks to achieve their goals. It is usually associated with individuals working together across an internet or intranet connection. These software systems are

introduced and analyzed in section 2.5.2. They can range from ordinary text chats to complex visualization and decision support tools.

Even though, with current information technologies, the software infrastructure allows the conceptual design process to be fully decoupled from the physical presence of the stakeholders, the CE design process is still accomplished through a series of (mostly) local working sessions mainly for an effective team management (Figure 3). Each session can be used to organize a collaborative task, and the designers taking part in the session can share the design information dynamically (Rodriguez & A. Al-Ashaab 2005; Bidarra et al. 2002).

Figure 3: CE session at the Concurrent Design Facility of the European Space Agency (ESA - CDF 2011)

During a session, designers can have different roles and responsibilities, like on practical working groups. For instance, a designer can be a project leader or an expert in any of its smaller parts. A project leader is responsible for managing the session and keeping track of the design process evolution. An expert is responsible for a certain part of the system, often referred to as domain or subsystem, and should model, discuss, dynamically adapt, and decide on the design parameter assignments related to its area of expertise.

The other concepts introduced in the research motivation and objectives relate to an integrated conceptual design methodology (process) and two design support tools (software infrastructure).

A methodology can be thought of as a «recipe» of processes, methods, and tools to a class of problems that exhibit common characteristics which, in this case is an *integrated conceptual design methodology* to apply in complex engineering systems. The methodology proposed by this work focuses on solving the shortcomings of existing approaches (detailed in Chapters 2 and 3) by suggesting new methods to enable the representation of both the static structure and dynamic behavior of complex engineering systems without the complexity of formal modeling methodologies (§2.4). It introduces new visualization methods which can

make explicit and traceable the changes in the model, and facilitate the documentation of every decision, with a clear map between the system representation and the actual design process. Contrary to existing models, the proposed methodology is able to depict the status of the subsystem modeling, its relationships, and their impact on the overall system. It is a method to aid not only project managers (or system engineers) but also the different subsystem specialists involved in the design phase.

The proposed methodology details a set of diverse support tools contributing to a more effective and efficient design process. As an example, this work focused on a *decision support tool* that explores the idea of using the prediction capabilities of neural networks to provide suggestions for early design decisions based on previous solutions (Chapter 5). By abstracting the complexity of the model from the designer's perspective, this tool intends to cut on the time spent on individual modeling analysis during concurrent design sessions, and provide a quantitative metric of the expected error in order to ascertain the quality of the design prediction.

The other tool developed in this thesis was the *INtegrated Concurrent Realtime EnvironMENT* (INCREMENT) corresponding to the implementation of the main ideas behind the integrated conceptual design methodology. INCREMENT provides a software infrastructure to offer an integrated subsystem-centric modeling environment with a global system overview. It drastically increases the system's insight by offering a clear visualization of the design evolution, easing the decision support and documentation tasks, and providing a metric about the impact of decisions on other subsystems. The development of this tool is detailed in Chapter 5.

There are two common denominators amongst these support tools: visualization methods and knowledge management. These are currently keywords in numerous research works but, usually, the proposed methodologies in these fields are either too general or presented as silver bullet solutions (Y. J. Chen et al. 2008)(§2.2 and §2.3). This work recognizes the complexity of these two concepts, and does not offer restrictive static applications; instead, it proposes a series of guidelines and their dynamic tools to ease the integration of these concepts at the conceptual design phase of complex engineering systems (Chapter 4).

The *Visualization Methods* (VM) approached by this work belong to the information visualization field which relates to the interdisciplinary study of "the visual representation of large-scale collections of information" (Bederson 2003). VM take advantage of the "human eye's broad bandwidth pathway into the mind to allow designers to see, explore,

and understand large amounts of information at once" (J. Thomas & Cook 2005). Instead of providing a static vision of the system in the context of conceptual design, the objective is to maximize human capacity to perceive, understand, and reason about complex and dynamic system behaviors. It builds upon the recognition of the reasoning behind the design process, as well as the underlying cognitive issues, to propose a series of perspectives that can maximize the designer's insight into the behavior of the system.

The goal is to enable fast high-quality human judgments by identifying critical information during the design process, and proposing several viewpoints consisting on a set of rules on how to display information during those distinct phases (Chapter 4).

The concept of *design information* is critical to enable informed decisions about the applicability of different VM to the different design phases. In the context of this work, design information is considered to be the important information about the system that must be transferred from one person to another (different designers) throughout the conceptual design phase. This work supports the same two key types of information identified by Vollertum and Igenbergs (2001): structure information and attributes.

Structure information describes which elements exist in the system and how they are connected by relations. This is also known as system architecture. Attribute (non-structural) information can be subdivided into: functional information about an element (internal functions the element can provide), and property information (values of design parameters). Usually not all pieces of design information are accessible from the same viewpoint. It is, therefore, vital to support several VM focusing on specific architectural representations and/or subsystems.

It is not by chance that there is an expression stating that «knowledge is power» and not «information is power». The web and IT have turned access to information into a ubiquitous task. We are currently passing from an information age to a knowledge age (Liao 2003), and the problem is not to obtain the information but to ascertain which of this information can support value-added tasks, commonly known as knowledge. *Knowledge Management* (KM) is the research field which focuses on developing a wide range of technologies and applications to deal with knowledge for both academic research and practical use. This work focuses on KM during the conceptual design by exploring the nature of knowledge during the design process, possible frameworks, architectures, methodologies and tools (§2.3). KM recognizes three types of learning during the design process: individual learning, learning through direct communication, and learning using a

knowledge repository (Van Heijst et al. 1997). Wiig (1994) considers KM as a human-centered activity that attempts to enhance these learning processes by using computer systems. Several guidelines are introduced on Chapter 4 and implemented in tools described on Chapters 5 and 6.

This work also approaches a series of concepts that, even though are not part of the main focus, are used in the proposed methodologies and tools:

- *multiple attribute* refers to the broad variety of metrics upon which an individual makes a decision, and in this case a design decision.
- *tradespace* is the full range of possible concepts that could define the design. These are the things that the designer can manipulate to develop a new design. The expansion of this tradespace is the essence of innovation, a creative recombination of current resources or systems to create a new system, which never before existed.
- *optimization* is applied in two distinct contexts in scope of this work: design process optimization, and system optimization. *Design process optimization* consists of cutting on the non-value added time wasted during the design process by rescheduling the design tasks, reducing the time spent during individual subsystem analysis and/or expensive computer calculations. *System optimization* focuses on choosing the best possible design solution to maximize (or minimize) a certain objective within given constraints or requirements. The system optimization can be achieved by various strategies such as tradespace exploration (Ross et al. 2004), or Multidisciplinary Design Optimization. The *tradespace exploration strategy* consists of looking at all the possible solutions and converging towards a final solution by means of informed decisions. The *Multidisciplinary Design Optimization* (MDO) (Pedregal 2003) focuses on finding design point solutions by formulating the problem as interacting subsystems with well-defined models, each possessing a certain degree of autonomy, but depending on other subsystems via a number of couplings, also known as interdisciplinary variables. This work supports the fact that the choice of the depth of the MDO strategy chosen should depend on the accuracy of the underlying models and the desired level of detail at the conceptual design phase.

With the previous definitions and contextualization, it is possible to formalize the earlier research motivation into the following research question: *In the context of complex engineering systems during the conceptual design phase, can the overall design solution and/or design process be*

enhanced by using the prediction capabilities of neural networks for decision support, and establishing a computer-based integrated design environment?

1.3. Thesis organization

The organization of the thesis is as follows. Chapter 2 focuses on the precursor work accomplished in the research fields related to the thesis topic. Using the definitions of the concepts introduced in Chapter 1, Chapter 2 provides a comprehensive critical review of the relevant work in the literature.

Chapter 3 concentrates on presenting the ideas that can contribute to solve some of the problems identified in Chapter 2. This chapter describes the results of a survey made to conceptual designers, maps the existing shortcomings in the field to possible solutions, and describes the research approach that led to the definition of the contributions of this thesis.

Chapter 4 proposes a new conceptual design methodology for complex engineering systems. The different methods and ideas are described in several stages, supported by examples. The tools described in Chapters 5 and 6 are part of this methodology.

Chapter 5 provides a short introduction to the theory of neural networks and describes the architectural choices that led to the development and implementation of a decision support tool. This goal of this tool is to reduce the turnaround time during design iterations by cutting on the time required for modeling non-deterministic behaviors without compromising the dynamic evolution of the design. The performance gain is measured through a set of case studies involving possible decisions that need to be made during the conceptual design phase.

Chapter 6 details the implementation of INCREMENT, the tool enabling the proposed integrated design methodology. This chapter describes the architecture of the software, and introduces its main functionalities with the help of a potential design session of a telecommunications satellite.

Final remarks, conclusions and future work can be found in Chapter 7. Figure 4 shows the information flow through the general structure of the thesis.

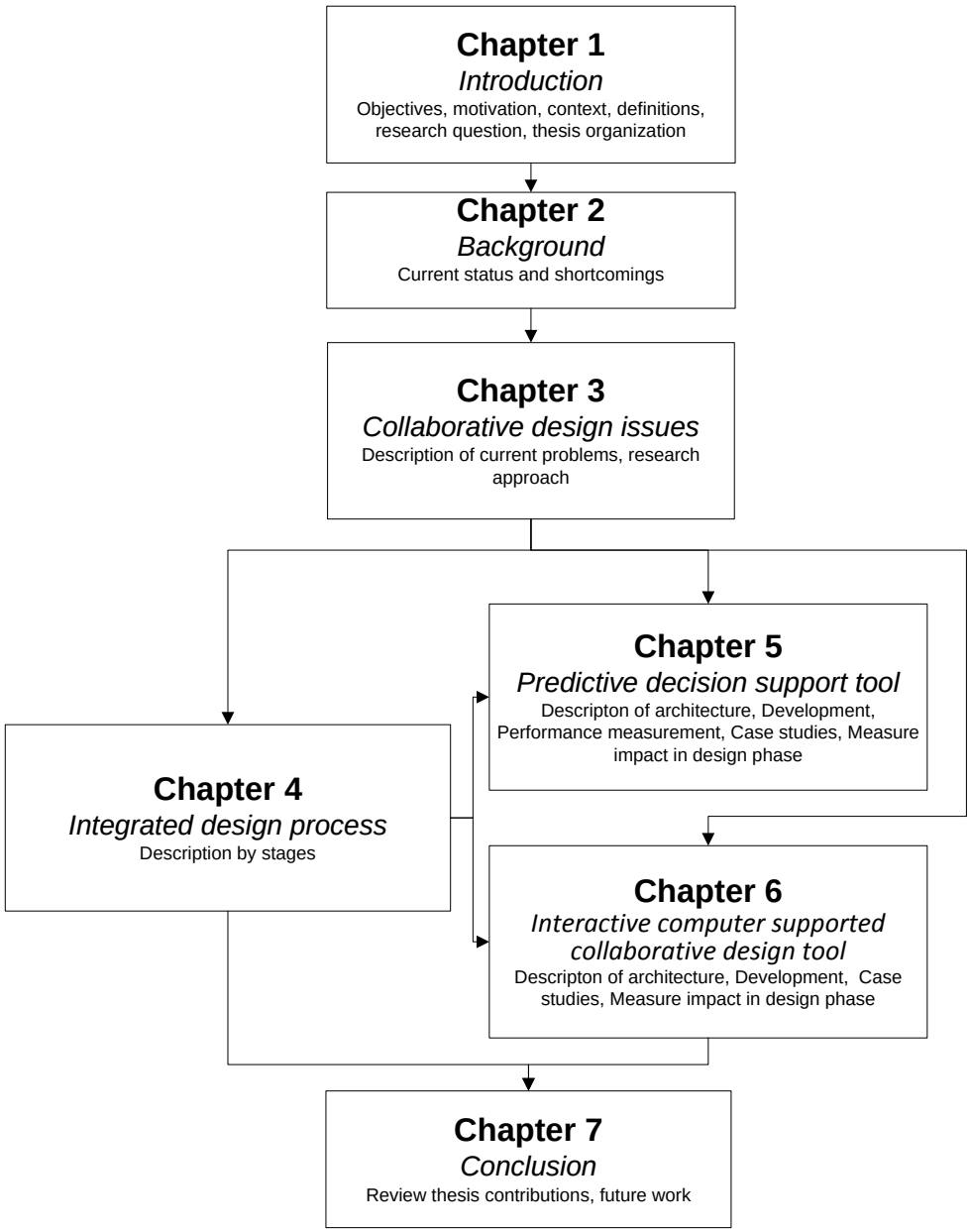


Figure 4: Thesis structure

[*This page intentionally left blank*]

2. BACKGROUND

During the past decades, complex engineering systems have experienced some major technological innovations and paradigm shifts (W. D. Li & Qiu 2006). To meet the increasing demands of global markets, the most recent research is aimed at updating existing systems in order to make them collaborative and distributed.

Collaboration during the design process can be organized in either a horizontal or a vertical mode (Molina et al. 1995). The horizontal collaboration puts the emphasis on gathering a design team from the same or different disciplines to carry out a task systematically. The vertical collaboration focuses on establishing an effective communication channel between the upstream design and the downstream lifecycle activities like production or maintenance.

The scope of a CE-based collaborative environment is to incorporate both the vertical and the horizontal collaboration modes, support distributed applications, and consider different subsystem models real-time (synchronous collaboration). A CE-based collaborative environment involves the integration of very diverse tools in terms of functionalities, communication protocols, programming languages and data structure representations.

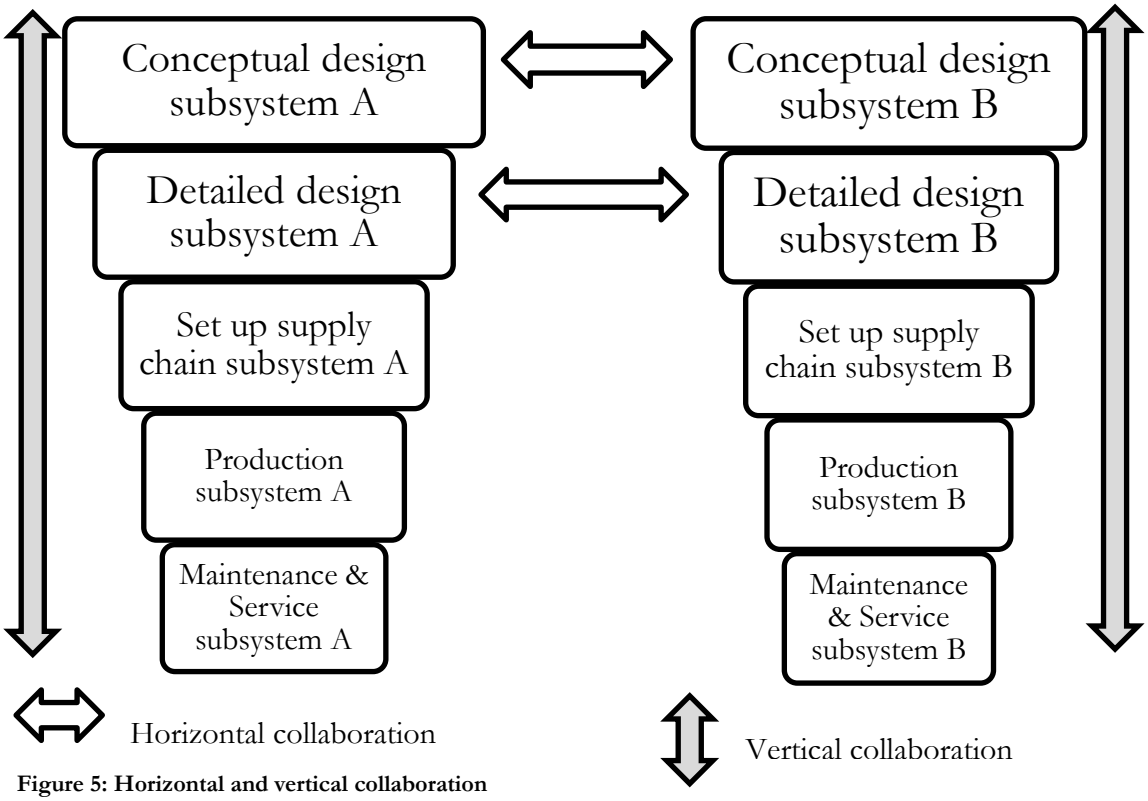


Figure 5: Horizontal and vertical collaboration

Concurrent Engineering had its debut around 1980s, making it a relatively new paradigm, but has had the opportunity to mature in recent years to become a well-defined systems' approach towards optimizing the design process. CE aims to speed up the design of systems by involving all the relevant players from the beginning of the development of the system or product in the same collaborative environment. It is typically used in the early stages of design, but some recent studies also start to shown a trend to use CE at more detailed design phases (Findlay et al. 2011). CE removes the physical and organizational boundaries to communication, so that design tasks that once took months or even years, can be completed in a matter of days or weeks. Real-time interaction (synchronous collaboration) speeds up the process by promoting the early resolution of design issues and clearing up misunderstandings with face-to-face communication (Nevins 1989).

Designing a system in a CE environment is not a straightforward task, especially if the system is complex. There is a general trend in literature supporting the view that the performance of this process can increase by using tools that can be easily accessed and that coherently and dynamically represent the system, contributing to make the design more effective and efficient (Vira 1983; Bresciani & Eppler 2009a; Du & W. Chen 2000).

Although there is a generalized idea that CE can considerably gain by an increased use of IT to automate at least part of the process, it is still mainly an *ad-hoc* process with too much time spent on subsystem level analyses and interaction problems, and not enough on design iteration (Tomiyama 2006; Reddy et al. 1993). This leads to the major «perceived» disadvantage of CE: that it increases the time spent in preliminary design when the designers are anxious to finalize details and release a conceptual solution. The methodology and tools proposed in this work have the goal to increase both the «real», and the «perceived» performance of CE.

Recent research works have focused on the discovery of the general failures that can occur when interacting with complex problems in an environment such as CE (Lindemann et al. 2009). Table 1 shows the main results of these findings.

Table 1: Main failures and causes when interacting with complexity (Lindemann et al. 2009)

Failures when interacting with complexity	Causes
Tendency to encapsulation (endless planning and information acquisition)	**Protection of one's own competence**
Assimilation of situations to already known situations	
Solving of only known problems	

Immediate acting instead of detailing of objectives balancing of contradictory objectives and determining the main focus	**Slowness of thinking**
Reduction to one central parameter	
Determination of general rules	
Abstraction	
Simplification of temporary processes (e.g., by linear extrapolation)	
No consideration of side effects	
Methodism	
«Ballistic decisions» (neglecting resulting effects)	
Ad-hoc reactions (only considering the actual situation)	**Minimal recording of information**
Neglecting implicit problems	**Fixation of attention to the actual problem only**

CE is based on the three elements introduced in section 1.2: process**,** multidisciplinary team**,** and software infrastructure. These elements share commonalities with five well-established research fields:

- the study of *design processes***,** with the goal of establishing guidelines on the main actions that will guide the conceptual design phase. A design process aids vertical collaboration and is the backbone of every system's design phase. It defines the direction of information flow along with the necessary deliverables within a certain project management strategy (§2.1),
- *visualization of complex systems*, which is field focusing on applying VM to increase the insight into the behavior of complex systems. It also approaches some applications dedicated to enabling collaborative and distributed design or product preview/review (§2.2),
- *knowledge management* (KM)*,* which focuses on the development of methods and tools to help capturing, organizing, generating and distributing knowledge within projects or organizations. KM efforts have well established applications in the information sciences and business fields (K. Wiig 1994), but more recent applications have also emerged in the product lifecycle and design fields (Thouvenin et al. 2005) (§2.3),

- *design modeling methods*, which focus on the establishment of a common language to represent the system's behavior in a coherent and detailed way throughout the design phase (§2.4),
- *computer supported collaborative design* (CSCD), which concentrates on the application of collaborative engineering to product design by studying possible modeling methodologies, computer-aided decision support, and IT-enabled tools (§2.5).

The following sections approach the precursor work made on these fields, introducing the main achievements and limitations regarding a possible application during the conceptual design of concurrent engineering systems.

2.1. Design processes

The process of designing a system has historically been seen in two particularly different ways (Dorst 1995) both of which are referred by several names in literature. One of the paradigms is known as the rational model (Brooks 2010), technical problem solving (Schon 2000), or the reason-centric perspective (Ralph 2010). The other paradigm is known as the action-centric perspective (Ralph 2010), reflection-in-action (Schon 2000), or co-evolution (Newell 1972).

The rational model considers that the design process starts with a set of known objectives and constraints, and that the role of the designers is to optimize a design candidate to comply with those desired characteristics. The design process is considered as a series of tasks informed by research and knowledge with well-defined procedures (guided by a set of rules) and a plan with several stages (Brooks 2010).

The action-centric perspective considers that creativity is the main driver of the design process and should be extensively exploited to generate the different design candidates. It points out that there are neither clear guidelines nor an apparent sequence of design tasks (Ralph 2010). The quality of the design critically depends on the talent of the designers and their dynamic as a team.

Both models exhibit limitations that have been identified in the literature. Extensive empirical evidence has shown that designers cannot design in an entirely rational way (Cross 1992), which undermines the main premise of the rational model. Another problem with the rational model lies in the fact that design objectives are often unknown when the

design process begins, and the requirements and constraints continue to change throughout the design process. This is demonstrated by the famous quote by Henry Ford: "If I'd asked customers what they wanted, they would have said a faster horse". The same idea applies to designing an engineering system, if the designers must restrain themselves to a set of objectives and constraints, the design solutions will be nothing more than incremental. The action-centric perspective is much more complex than the rational model, slower and is, therefore, intuitively perceived as a less performing option. Yet, due to the incorporation of innovation into the design process, it can lead to a design solution with a higher overall performance.

In engineering, design is commonly seen as a part of a larger engineering process consisting of "the application of scientific and mathematical principles to practical ends such as the design, manufacture, and operation of efficient and economic structures, machines, processes, and systems" (Ertas 1996). It is mainly seen as a problem solving activity guided by mathematical and scientific principles, a perfect application for the rational model.

However, engineering systems are becoming more and more complex, not only because of their intrinsic nature of serving more difficult and broader goals, including minimizing the cost, but also because of other multidisciplinary factors that have to be considered such as sustainability, logistics or environmental issues. The market pressure for more innovative designs rather than incremental ones have raised the importance of more «human-centered» design processes such as the action-centric perspective. The correct choice of a design process for a specific system lies in the balance between the rational and the auto-centric models. This is one of the objectives of Systems Engineering (SE), which is the interdisciplinary field of engineering that focuses on how complex engineering systems should be designed and managed over their lifecycles (Laudon 2011). When dealing with the design phase, SE proposes three main design processes: the waterfall, the V-Model, and the spiral (W. D. Li & Qiu 2006). The first two are rational models while the last one is an action-centric perspective.

The *waterfall* is a classical linear and sequential design process that originated in the manufacturing and construction industries (McConnell 1996). In this process, the different stages are clearly divided and are typically assigned to separate teams to ensure greater project and deadline control. It considers that each stage is 100% complete and entirely correct before proceeding to the next phase (pure waterfall model), or only allows iteration

of one stage with the stages immediately upstream or downstream (modified waterfall model - Figure 6). Its main criticism arises from the fact that it is impossible, for any non-trivial system, to have one of its design stages perfectly «closed» before moving to the next phases (Parnas 2003). The requirements typically change during the design process and the system must be modified to accommodate the new constraints.

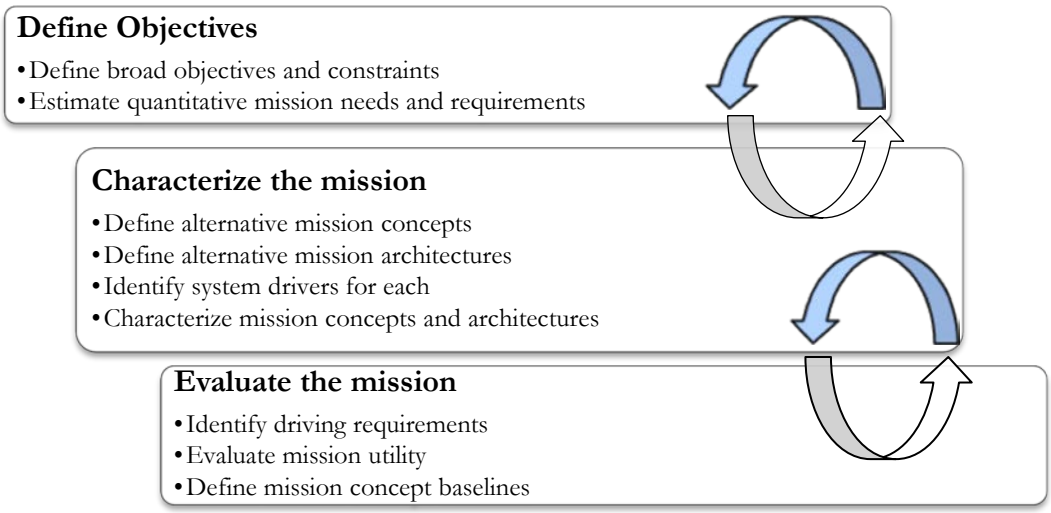


Figure 6: Example of a modified waterfall design process applied to the design of space missions (Larson 1997)

The *V-Model* is a design process that aims to improve control by specifying the expected results and roles during the design phase. It establishes a notion of quality control to make sure that the aimed results are accurate and have the desired quality (Estefan 2007). Each design team is involved in at least a definition and a developing activity (Figure 7). Standardized interim deliverables are used to justify design decisions as the design process advances. The main disadvantage of the V-Model is its rigidity to reiterations.

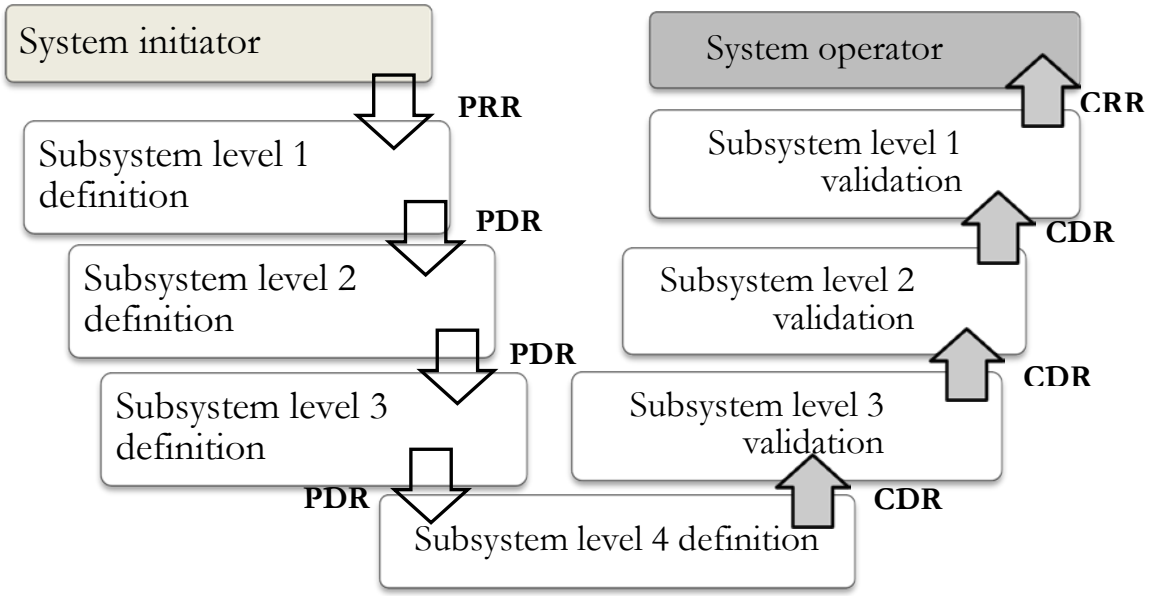


Figure 7: Typical V-Model with interim deliverables: Preliminary Requirements Review (PRR), Preliminary Design Review (PDR), Critical Design Review (CDR) and Commissioning Result Review (CRR)

The *spiral* design process is based on the continuous refinement of requirements, analysis, subsystem design, and design verification. It is part of an agile process (Larman & Basili 2003) supporting an iterative and incremental approach to enable a swift detection of environmental changes and new techniques, and customization of systems to accommodate changing requirements (Reich et al. 1999). In a spiral design process, preliminary solutions evolve through the different subsystems and preliminary designs and converge towards a final solution. Its main drawback is inherited from its action-centric perspective, using a spiral process at the conceptual design phase makes it a slower task and is therefore perceived, from the designer's perspective, as less efficient.

The design process in a CE environment is often associated with the spiral design process (Bandecchi et al. 2000) where the overall system requirements evolve towards more detailed requirements, preliminary conceptual analysis towards more detailed verifications and, high level design decisions give way to lower level decisions (Figure 8).

One of the most important reasons for the success of CE is that it chooses to use an iterative development method as opposed to a sequential one, like the waterfall, or the V-Model. By not looking at the design process in an entirely linear way, it allows the inclusion of creativity and makes sure the design is not static and able to evolve in any direction at any stage (Hartley 1992).

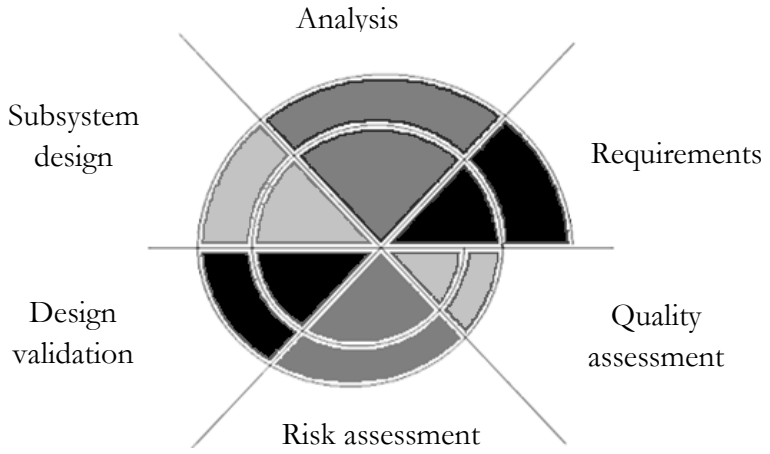


Figure 8: The Concurrent spiral design process

The CE design process increases the pace of conceptual design by bringing together all relevant personnel to conduct focused collaborative studies. In contrast with the traditional rational models, CE allows the implementation of a spiral process by explicitly removing the physical and organizational boundaries to communication so that design tasks that once took months or even years to accomplish can be completed in a matter of weeks (Dowlatshahi 1994). Real-time interaction (synchronous collaboration) between specialists

enables an accurate dialogue that resolves issues immediately. Getting everyone, including the customer, together in the same place not only speeds up the process but also offers the participants the ability to clear up misunderstandings with face-to-face communication. Each member of the team has the responsibility, from the beginning of the design process, to lend expertise to develop the best possible product within given cost, and schedule constraints.

2.2. Visualization of complex systems

There is an Indian tale, which refers to a group of blind men who touch an elephant to learn what it is like. Each one touches a different part, but only one part, such as the side or the tusk. They then compare notes on what they felt, and learn they are in complete disagreement. The blind man who feels a leg says the elephant is like a pillar; the one who feels the tail says the elephant is like a rope; the one who feels the trunk says the elephant is like a tree branch; the one who feels the ear says the elephant is like a hand fan; the one who feels the belly says the elephant is like a wall; and the one who feels the tusk says the elephant is like a solid pipe. A wise man explains them: "All of you are right. The reason every one of you is telling it differently is because each one of you touched a different part of the elephant. So, actually the elephant has all the features you mentioned". A complex system, just like the story, exhibits different characteristics when seen from different perspectives. Only by having an idea of the different viewpoints one can see the total system or the elephant.

Despite the apparent advantages and recent advances of visualization in engineering design and optimization, many consider the state-of-the-art in visualization to be still in its infancy. Appropriate representations (i.e., visualization and manipulation strategies) can already be used to better understand the models, algorithms, data, and design candidates obtained during the design process (Jones 1994), but a broad discussion is needed to study the use of visualization as part of the solution during the whole process rather than just a way of presenting results (Messac & X. Chen 2000).

Visualizing a system critically depends on the form of its underlying structure, it may be organized into a tree, a ring, a set of clusters, or some other kind of configuration, and it is up to the designer to decide which of these forms is best by also considering the aspects he/she wants to emphasize. Literature diverges on the necessary path to discover the underlying structure of a set of entities. While some consider this discovery as an art, highly

dependent on the talent of the individual (Tufte 2006; J. Thomas & Cook 2005), others argue that the choice of the possible VM for a specific application is a science with well-established rules (visual-literacy.org n.d.; smashingmagazine.com n.d.; Kemp & J. B. Tenenbaum 2008).

Usually the design process of a complex engineering system does not begin with all the information needed to develop its correct representation at different architectural levels (functional, physical, technical, dynamic operational, and informational), and the developer may easily become overwhelmed with the initial disorder of ideas (Loureiro & Leaney 2003).

The examples of VM and their applications are numerous in the literature (e.g. smashingmagazine.com n.d.; Heer et al. 2005; Lanza 2003; Panchal et al. 2007; Kemp & J. B. Tenenbaum 2008). The VM can range from basic data models (Figure 9) representing quantitative data in schematic form, to more interactive and complex information models (Figure 10). VM are nowadays easily accessible to the global community through the documentation and classification efforts of avid web-based communities such as (visualcomplexity.com n.d.) or (visual-literacy.org n.d.).

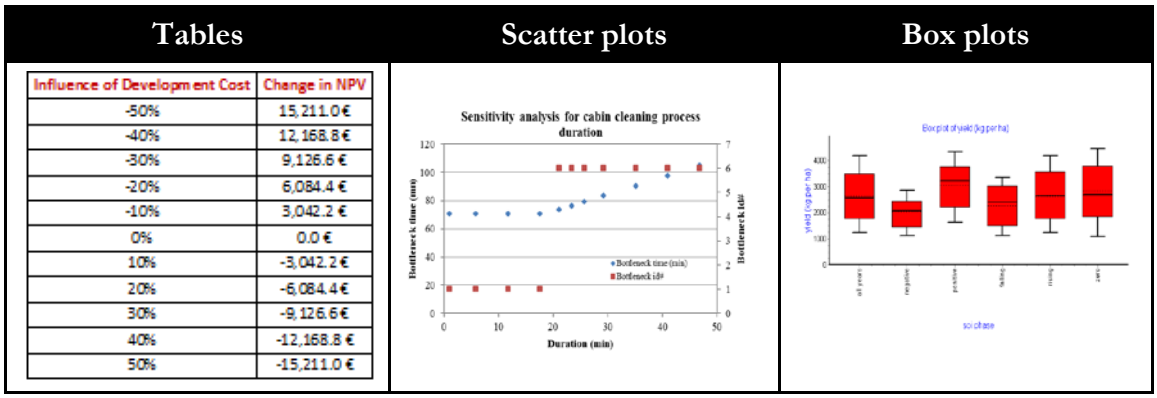


Figure 9: Example of traditional data VM

The issue of visualizing complex engineering systems at the conceptual phase is mentioned as an ill-defined problem in visualization, because the complete definition of the model and/or of the objectives is dynamic and, therefore, impossible to freeze in time. At the beginning, the designer may be more interested in visualizing the requirements traceability, but at the end, he/she might want to know the specific distribution of a system parameter such as mass. As the model and the objective specification becomes less precise, visualization becomes more important (Eddy & Mockus 1995). It is necessary to review the possible VM and identify (i) the most relevant VM for complex engineering systems, capable of doing (if possible) all the desired support operations, and (ii) the VM that are equivalent, to avoid unnecessary repetitions and implementation effort.

Several companies have recently tried to put these VM into use in the context of complex engineering systems and have developed their own platforms, such as Oracle AutoVue (AutoVue 2011), Actify SpinFire (Spinfire 2011), SolidWorks eDrawings (eDrawings 2011), RealityWave ConceptStation (ConceptStation 2011), and Autodesk Streamline (Streamline 2011).

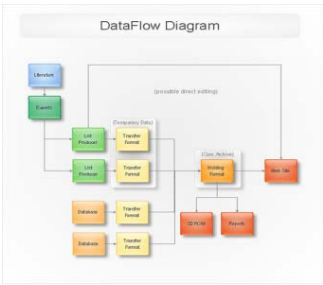
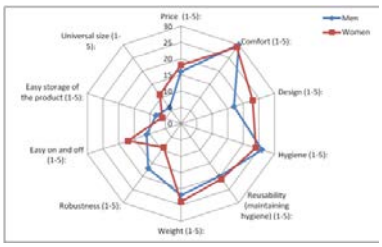
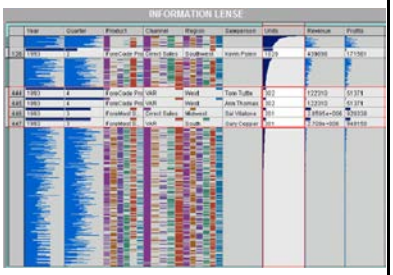
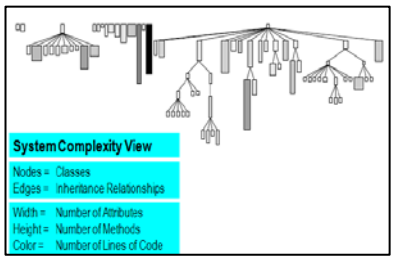
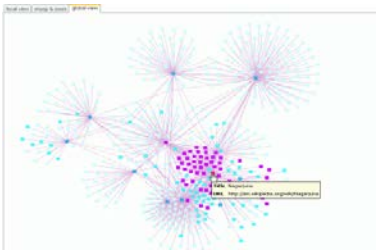
Data flow diagrams	Radar charts	Information lenses
		
Data flow diagrams allow the designer to gain insight into the information flow inside the system, between the different subsystems or parts.	*Radar charts* allow the designer to have a global view on the performance of the different system configurations, having different parameters compared simultaneously.	*Information lenses* allow the designer to quickly grasp trends for a quantitative table.
Treemaps	**Polymetric graphs**	**Clustering graphs**
		
Treemaps display hierarchical (tree-structured) data as a set of nested rectangles. It can be used to put the impacts of each subsystem in perspective.	*Polymetric graphs* allow the designer to get an impression of the size, structure, and complexity of the system in terms of parts and inheritances. It allows locating important hierarchies, showing if there are any deep, nested hierarchies.	*Clustering graphs* allow the designer to «observe» the system from different views, e.g. a global view (where only the main subsystems are represented), a «zoom view» (where only directly-related subsystems are displayed), or a local view (where only a single subsystem is represented in detail).

Figure 10: Examples of information VM (visual-literacy.org n.d.; visualcomplexity.com n.d.)

These recent platforms are seen as useful solutions to approach the geometric design definition by using 3D formats for Web applications, such as the Virtual Reality Modeling Language (VRML) (Omar et al. 2009), or document the whole system lifecycle by allowing the design teams to collaborate more effectively, documenting design discussions, reviewing new requirements, etc.

Unfortunately, they are not tailored to approach the conceptual design, where the detailed models are scarce or even inexistent because they require thorough analysis that can only be performed during the later design phases.

Despite the remarkable number of publications about the benefits of using visualization in a variety of fields, from efficient project management to depicting pattern from large astronomical datasets (J. L. Rodgers et al. 1999; smashingmagazine.com n.d.; J. Thomas & Cook 2005; Bederson 2003), so far only a few studies have investigated the possible problems of using VM for communication or reasoning (Bresciani & Eppler 2009b).

These VM-induced pitfalls have generally been divided into two groups: designer-induced or cognitive user-induced (Tufte 2006; Wainer 2004). Designer-induced pitfalls are the most common, and can be related to: ambiguity (visual notations may contain unlabeled symbols that may be ambiguous and thus difficult to interpret), confusion (VM that do not have a clear overall logic), hiding/obscuring messages (VM may hide important insights contained in data by the way that data is represented graphically), misleading meanings (some VM are drawn in a way that may lead to incorrect conclusions), or even unclear goals (may leave too much room for interpretation regarding its purpose or main message). Cognitive user-induced pitfalls typically arise when the VM are difficult to understand because they depict many complex relationships that may not be optimally represented, e.g. representing all the parameter dependencies of a system in the same viewpoint without a clustering strategy.

A successful VM-based collaboration platform for the conceptual design of complex engineering systems should not only allow designers to get a quick general perspective of the system but also allow the display of how the different components of the systems, parts, and subsystems interact. It should be used in a CE environment and, therefore, needs to be dynamic, iterative, and creativity-enabling. It should guide to possible viewpoints that can be used by the different designers but still leave some room for the designer's cognitive tasks. It should accept that, at the conceptual stage, the design models are not mature enough to use only one type of viewpoints and, instead, guide the design process to develop these same design models. Finally, it should avoid the known limitations and hazards identified in literature and mentioned earlier.

2.3. Knowledge Management (KM)

The number of companies and systems that have recognized the importance of knowledge has been steadily increasing in the last decades (Desouza 2005). As we enter into the «knowledge age», this resource is not only seen as a way to preserve historic heritage or learn new competences but is also seen as critical to help solving problems and approaching new challenges. During the last decade, influential companies have accepted that competition is shifting towards how well knowledge and other intellectual assets are managed. This has led many of them to pursue strategies to actively and explicitly manage knowledge (K. M. Wiig 1997) and to recognize that competitive advantage is only gained by making individual knowledge, both current and past, available to the entire organization (Hicks et al. 2002). If KM is carried out in a swift way, new products can be brought to market earlier, business processes can continually be improved, and innovative new ideas brought to commercialization.

Indeed, the way a company or system deals with its knowledge is the key consideration when measuring its maturity. The common standard is the Capability Maturity Model (CMM) which is considered to be the most adopted aid in improving organizational business processes in diverse areas, such as software engineering, or project management. CMM defines the concept of maturity level as a well-defined evolutionary step toward achieving a mature process. Each maturity level provides a set of process goals to deal with knowledge. A company or system is level 1 if it is not able to reuse its knowledge, level 2 if it can reuse knowledge for repeatable projects, level 3 if there is a process to deal with knowledge, level 4 if knowledge is correctly managed and exploited for new projects, and level 5 if the set of processes to deal with knowledge are continuously optimized (Paulk 1995). Basically, the higher the CMM level of a company or system, the more efficient it is dealing with its knowledge, which translates into increased robustness to tackle new projects.

The issues concerning the definition of knowledge assets have been the scope of early work by Liebowitz and Wright (1999), and later formalization by Rowley (2008). The latter argues that assets can be created as sensory stimuli (signs); objective facts or observations (data); organized or structured data which has been processed in such a way that it now has relevance for a specific purpose or context (information); or information that has been processed, organized or structured in some way or applied into action (knowledge).

Different authors have proposed similar hierarchies with the purpose of organizing this intellectual capital according to their objective, complexity and level of detail, the most common are known as the «DIKW» (Data, Information, Knowledge and Wisdom) (Rowley 2007), and the knowledge pyramid methods (Zins 2007) (Figure 11). The main difference between these two lies in the definition of data as a signal (perceived by our senses), by the «DIKW» model, as opposed to a fact, by the knowledge pyramid. Also, what the «DIKW» model refers to as wisdom, the ability to increase the effectiveness by using judgment, is included in the definition of knowledge represented on the knowledge pyramid.

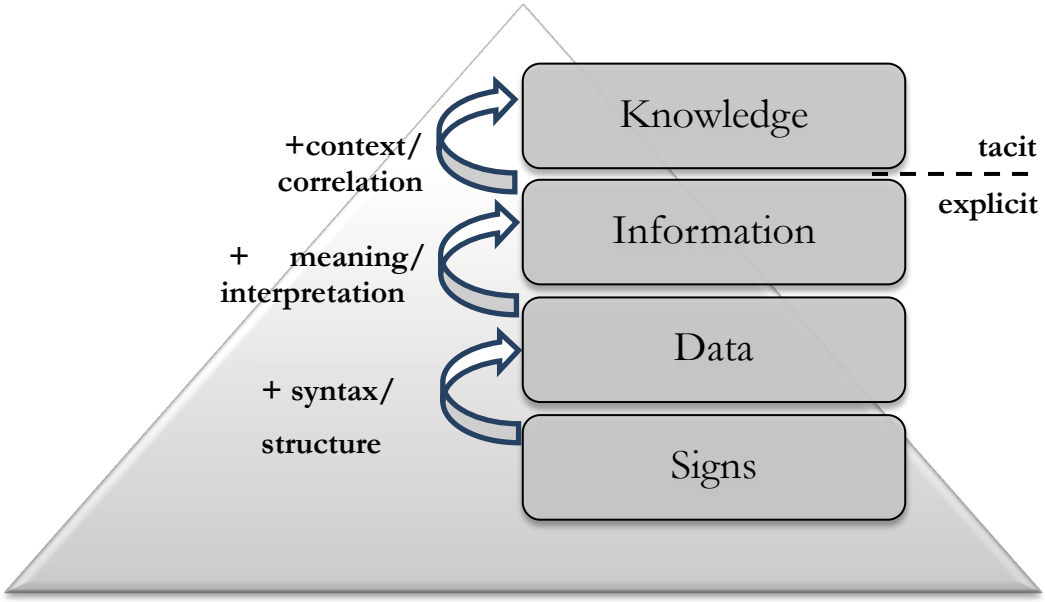


Figure 11: The knowledge pyramid, adapted from **(Zins 2007)**. Different levels from explicit to tacit knowledge

A clear distinction between explicit and tacit material is established in the literature. Explicit material relates to signs, data, or information that can easily be expressed, communicated, and shared in the form of hard data, scientific formulae, codified procedures, or universal principles. Tacit material refers to knowledge which is highly personal and hard to formalize such as complex relationships, subjective insights, intuitions and hunches.

The concept of KM was recognized as an academic discipline with the work of Nonaka (1995). In the beginning of 2011 there were over 20 distinct academic journals available related with KM, including some that approach the «pure» KM issues (Bontis et al. 2006). KM definitions are abundant in literature, although the general idea behind the concept is the same for every meaning, they usually focus on companies (not complex engineering systems), but are very different in their scope, some being more general such as "the process of collecting, organizing, classifying and, disseminating information throughout an

organization, so as to make it purposeful to those who need it" (Albert 1998), others more specific such as "analyzing information on a company's computer databases so this knowledge can be readily shared throughout a company, instead of languishing in the department where it was created, inaccessible to other employees" (Malhotra 2001).

A more useful definition is given by Argote (2000) defining it as "the systematic process that encompasses capturing knowledge in a way that helps people articulate it in an easily shared way, organizing it in a structured form that allows the standardization of its use, developing it by extracting the most useful information from previous studies, distributing it by helping and/or encouraging to (re)use knowledge" (Figure 12).

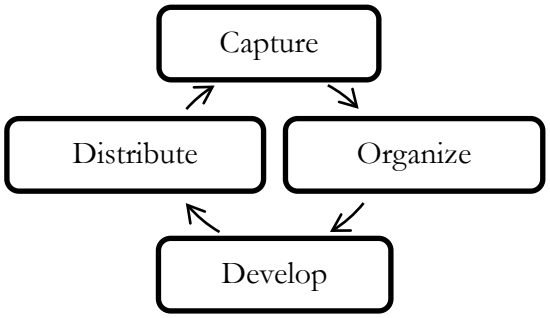


Figure 12: KM process

KM is achieved with the help of knowledge-based systems which range from: a knowledge repository, an inference engine, a knowledge engineering tool, or specific user interfaces (Dhaliwal & Benbasat 1996). These systems are supported by IT applications such as expert systems, rule-based systems, groupware, or database management systems (Laudon 2011). They focus on enabling three types of cognitive tasks: individual learning, learning through direct communication, and learning using a knowledge repository (Van Heijst et al. 1997).

One of the challenges in the design of complex system is to transform tacit knowledge locked in the years of expertise of the different stakeholders, into explicit information. It is not possible to achieve this by simply looking at the parts or features of the system. It requires acting upon it, changing it and observing the impact on the design (Kemp & J. B. Tenenbaum 2008).

A correct perception of the overall behavior of the system is especially critical during the conceptual design phase. Several authors have identified the common problems that arise when trying to implement KM in a CE environment: IT-based KM systems are still based on burdensome non-intuitive software (Liebowitz & Wright 1999); designers (and others) do not usually like to document things (Albert 1998); documentation efforts are typically

not recognized (Argote 2000); it is necessary to convince both engineers and management of the cultural, technical and organizational requirements to build a usable and useful KM strategy (King & Majchrzak 1996).

A critical issue when trying to establish a successful KM strategy is the common motivation related with the feeling of recognition, how one is perceived, and how can one better perceive the others. This fact has been made explicit through comprehensive experiments published by Thouvenin (2005) in the field of mechanical design and upstream modeling, showing that there is a clear correlation between the common motivation and the desire to collaborate.

Recent works focusing on interactions during the development of open source software on distributed web-based environments, such as Linux (linux.com n.d.) or Eclipse (eclipse.org n.d.) can provide some guidelines on how to increase the motivation of designers to use KM philosophies. There are strong parallels between software development and the design process of complex engineering system. In both cases there are several subsystem models, and numerous design parameters that need to be synthesized into a cohesive whole. Linus Torvalds, the founder of Linux was a pioneer on the style "release early and often, delegate everything you can, be open to the point of promiscuity". A survey conducted to the Linux community identified three categories of motivation (Edwards 2001):

- collective motives include identification of the developers with the goals of that community and the strong desire to help make those goals come true,
- peer recognition motives, not as important, include the wish to work together with friends and colleagues – and for those people to rank you highly,
- direct reward motives include learning new skills, making more money, or gaining new friends. Open source developers are usually not motivated by money; having fun and learning ranked highest in the survey.

Eclipse lead developers have emphasized meritocracy, transparency and openness to contribution from everyone as the three basic principles governing the Eclipse team's work ethic (Gloor 2006). All these advices should be considered as lessons learned for integrating KM in a CE environment.

In a CE environment, the specialists of the different subsystems tend to detach themselves from the global perspective in order to restrain the scope of their analysis, and define clear objectives for their work, creating a small «bubble» around their subsystem where their vision of the global system is simply manifested through their specific requirements and

interfaces (Kusiak 1993). This is often due to the typical systematic approach to a problem, if you cannot handle it, divide it in smaller (more comfortable) parts, and make the necessary simplifications to represent the other parts of the system that do not concern you directly. Unfortunately, this approach does not yield satisfactory results with complex engineering systems since they cannot be reduced to a sum of simpler parts that can easily be quantified and independently analyzed (Goldfinger 2000).

2.4. Design modeling methods

The design modeling methods are used to define the system during the design process. They establish a language to represent the system by enabling various levels of abstraction that can be lexical, graphic, or a combination of both. The foundation of system design modeling methods is to describe the perceived universe into two parts, the part «inside» the system and the part «outside» the system. From the «outside», the system receives inputs. To the "outside" the system delivers outputs (Chapman 1992).

The first contributions to this field arrived with the Entity Relationship Model (ERM) which introduced the first concepts of data modeling to the engineering community (P. P. Chen 1976). The ERM used only two concepts to describe a system: entities which were defined as "a thing or object of significance, whether real or imagined, about which information needs to be known or held" (P. P. Chen 1976) and relationships to qualify the interaction between these entities. Unfortunately, "the choice between an entity and a relationship proved to be completely subjective, showing that the same object can quite legitimately be regarded as an entity by some designers and a relationship by others" (Date 2004).

It is during the 1980s that the first methods started to emerge with multiple simultaneous representations of the system. The goal was to clarify the system model by considering the modularity of engineering systems to develop a modeling language based on its architectural decomposition (Sata et al. 1985). The fact that there usually is a hierarchy representing the different elements of a system, intuitively supported an architectural view using a top-down classification e.g. system, segment, element, subsystem, assembly, subassembly, and part (NASA 2007). One of such examples is the Structured Systems Analysis And Design Method (SSADM) (CCTA 2000) that included diagrams for logical data, data flow and entity behavior modeling.

In the 1990s there was already an abundance of modeling languages which posed a barrier for adoption from the industry's point of view (Date 2004). There were, for example, models used for describing the basic properties of the different objects of the system, solid models for shape representation, symbolic mathematical formulas, Finite Element Models (FEM) or other engineering models (Lindemann 2009). In 1993, Raddy focused on increasing the clarity and integration of the different models by establishing the division into product, process and organization models (PPO) (Reddy et al. 1993). This was the basis for other integrated product development methods (Loureiro & Leaney 2003) using a total view approach to manage the complexity associated, not only with a product itself but also with the interactions outside of it.

With the increased complexity of engineering design, the structural diagrams started to be insufficient to achieve a clear representation of the complete behavior of the system. Some authors realized that it was essential to combine the capability to keep track of both the static structure, and the dynamical behavior of the system by providing fine-grain version control of the elements (Cardei et al. 2008). This led to the first methods with the capability of representing the dynamic actions of the system by using behavior or interaction diagrams. Their research efforts resulted in a unified method, the Unified Modeling Language (UML).

UML is a standardized modeling language mainly used in the IT field for object-oriented software engineering. UML is a broad language that includes a set of graphic notation techniques to create diagrams such as data modeling (entity relationship diagrams), business modeling (work flows), object modeling, and component modeling (OMG n.d.). It is a mature language that has been evolving through an active team of developers since its creation. Although it is highly used in the IT-related industries, it is not specifically tailored for complex engineering systems which require a semantic foundation for modeling system requirements, behavior, structure, and system parameters (OMG n.d.).

The *SysML* (Systems Modeling Language) initiative originated in 2001 to customize UML and overcome the limitations found when approaching system engineering applications (Estefan 2007). SysML's semantics are more flexible, reduce UML's software-centric restrictions and add two new diagram types, requirement and parametric diagrams. Requirement diagrams are used to efficiently capture functional, performance and interface requirements, while parametric diagrams precisely define performance and quantitative constraints. SysML is now becoming standard practice in the industry, but it is considered a

complex language where symbols have different meanings depending on the type of diagram (Weilkiens 2007). It has a relatively long learning curve and, therefore, requires a reasonable amount of effort from the designers' point of view.

SysML is part of a new field that emerged during the last 10 years that focuses on modeling languages for design purposes, Model Based Systems Engineering (MBSE). It often overlaps with the notion of design model methodologies in providing an interdisciplinary approach for the successful development and realization of complex problems in many different disciplines. MBSE is a holistic field not specifically targeted on software-intensive systems. It represents a paradigm shift from a traditional waterfall design process to a model-based process (coherent with the spiral model) where activities that support the engineering process are to be accomplished through the development of increasing detailed models (Estefan 2007).

The goal of this work does not revolve around MBSE methods; there are already extensive lists for this purpose (Blanchard 2008; Kossiakoff 2003; Estefan 2007). The methodology proposed in Chapter 4 is based on key ideas from two MBSE methods: *Harmony-SE* and the *Object Process Methodology* (OPM) which are briefly introduced in sections 2.4.1 and 2.4.2. More detailed analysis on these two specific methods can be found in several organizations such as INCOSE (incose.org n.d.), or NASA (NASA 2007).

Another compact design method known as the *Design Structure Matrix* (DSM) is introduced in section 2.4.3**.** The DSM does not share the same holistic view of the system as MBSE methods but focuses on a major need in engineering design management, to document information that is exchanged. "It is a very proficient method in capturing, communicating, and organizing engineering design activities and architectural issues such as project team formation and product architecture" (Browning 2001).

2.4.1. Harmony-SE

The Harmony-SE methodology provides an "integrated, comprehensive sequential approach to: identify/derive required system functionality, identify associated system states and modes, and allocate system functionality/modes to a physical architecture" (Estefan 2007). It was created as a tool, and vendor-neutral model-based methodology, there are several tools supporting this method such as IBM's Rational Tau (ibm.com n.d.).

The main idea behind Harmony-SE is «service request-driven», the system structure is described by means of SysML structure diagrams using blocks as the basic structural

elements (Figure 13). Communication between blocks is based on messages (services requests). Provided services are at the receiving part of service requests and state/mode change or operations (activities) are described as operational contracts. Functional decomposition is handled through decomposition of activity operational contracts.

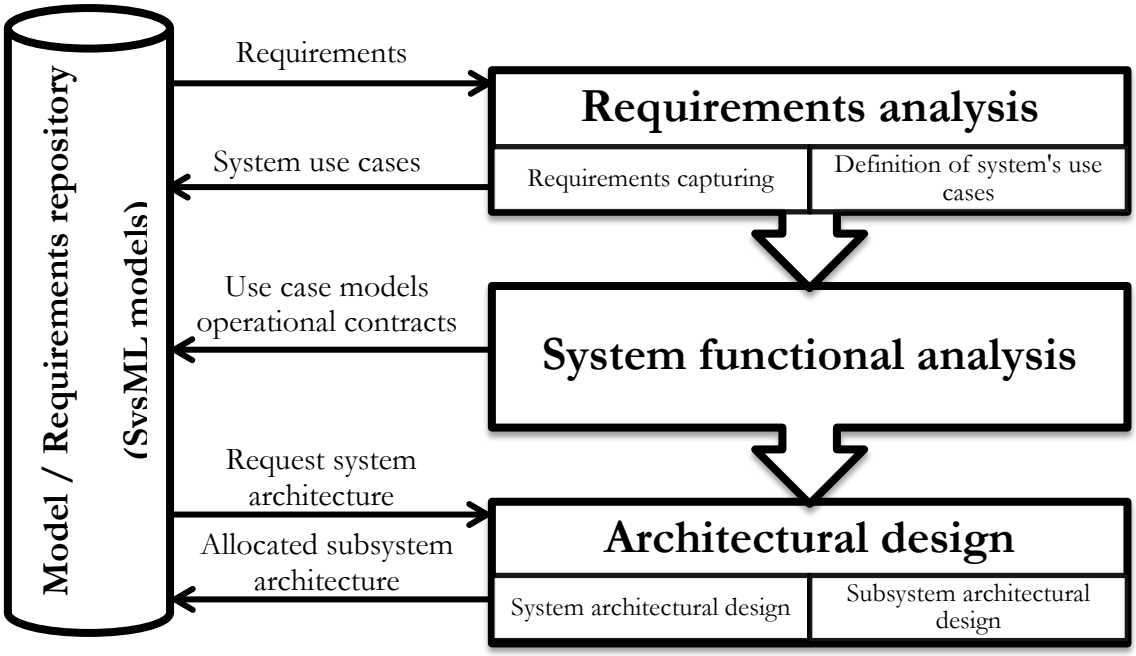


Figure 13: Request-driven Harmony-SE design method (adapted from (Estefan 2007))

Since Harmony-SE is based on SysML, it inherits all the limitations of this language mentioned earlier: complexity, contextualized (language depends on the type of diagram), and requiring a reasonable amount of effort from the designers' point of view.

2.4.2. OPM

The Object-Process Methodology (OPM) is a holistic approach for conceptual modeling of complex systems founded by Dov Dori in 2002. The OPM model "integrates the functional, structural, and behavioral aspects of a system in a single, unified view, expressed bi-modally in equivalent graphics and text with built-in refinement abstraction mechanism" (Dori 2002).

Two semantically equivalent modalities, one graphic and the other textual, jointly express the same OPM model. Each OPM element (entity or link) is denoted in symbols on a OPD (Object-Process Diagram), and natural language sentences known as Object-Process Language (OPL). This quest for unambiguity was also used in earlier methods such as the n-dim model (Reich et al. 1999) which used a universal modeling language that enabled functional, behavioral, and structural decomposition diagrams.

Every system consists of a set of objects, states and processes. Objects are the (physical or informatical) things in the system that exist, and if they are stateful (i.e., have states) then, at any point in time, they are at some state or in transition between states. Processes transform objects: they generate and consume objects, or affect stateful objects by changing their state (Figure 14).

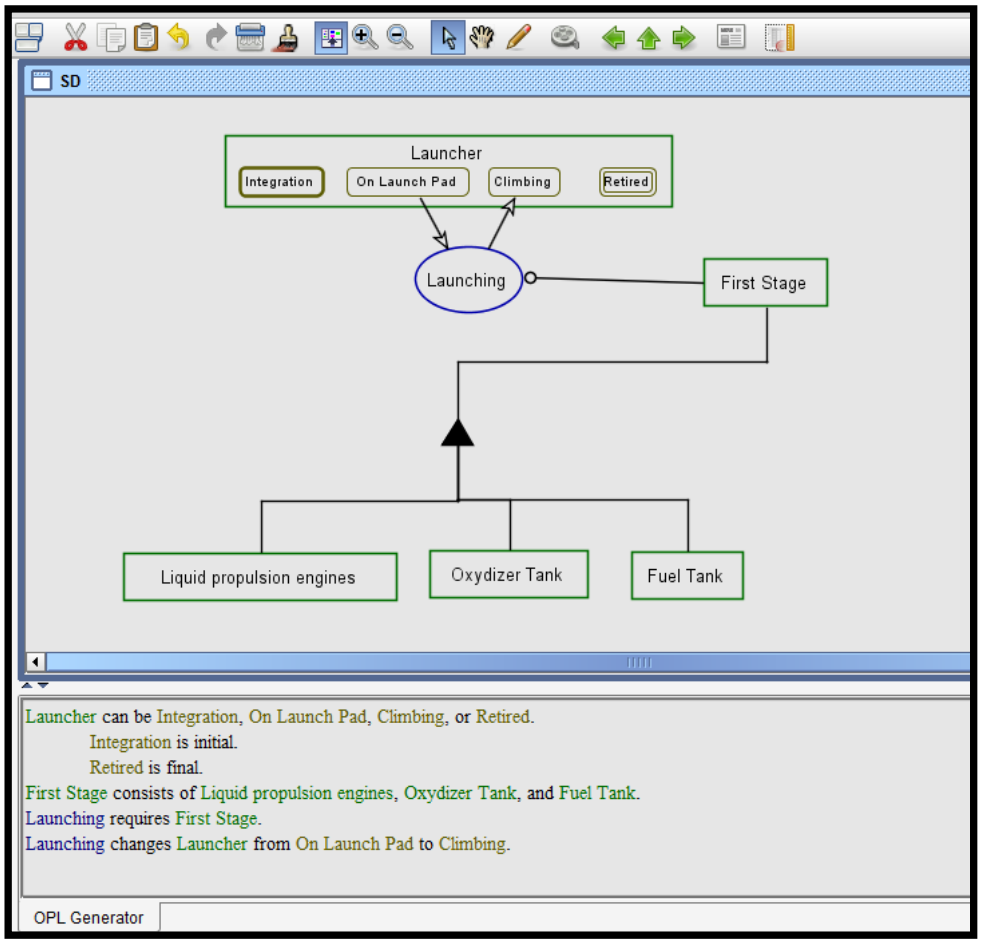


Figure 14: Example OPM model of a launcher (including OPD and OPL). Objects are represented as rectangles, states as circled rectangles, and processes as circles.

OPM exhibits a limitation at the requirements definition stage by not providing a suggestive way to transform requirements into a system architecture consisting of objects, states and processes. Despite that, it still is currently extensively used in industry applications through the OPCAT support tool (opcat.com n.d.).

2.4.3. Design Structure Matrix (DSM)

Browning argues that a better understanding of complex systems can be achieved by considering its structure (Browning 2001). He reasons that a procedure of dividing a system into subsets, capturing their mutual relationships in the form of inputs and outputs can increase its perception. Once the developers define an accessible system structure, it is even

possible to estimate the effect of some specific change on the overall behavior of the system.

Intuitively there are three basic types of relationships between the different elements of a system (Browning 2001):

- if a design item requires information from another subsystem, they are referred as being dependent and the information flow occurs sequentially,
- if they can be analyzed in parallel without requiring information from each other, they are referred as being independent and can potentially be analyzed at the same time,
- if, in order to be successfully analyzed, both design items require information from each other, then they are referred as interdependent or coupled.

These three types of relationships can be represented with the help of a Design Structure Matrix (DSM). The DSM is an n-square matrix representation of the system where the design elements are represented in the diagonal and their relationships in the off-diagonal entries.

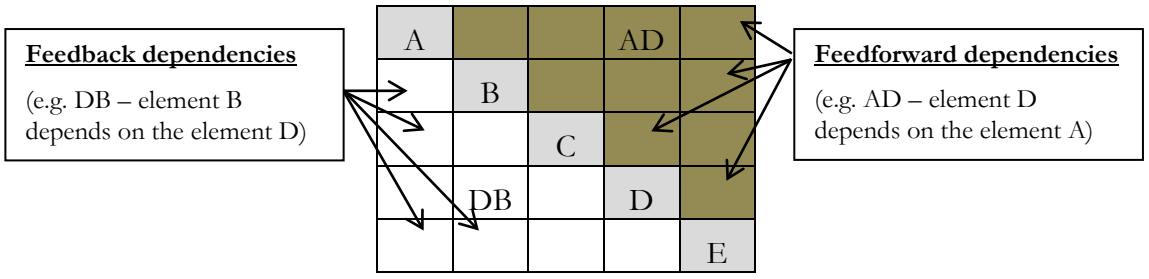


Figure 15: DSM description

There are innumerous examples in literature showing the application of DSM to model: activity dependencies and information flows (S. Eppinger 2001), transfer of documents and information (Yassine 2006), or schedules and cost distributions for the execution of planned tasks (Browning 2001; S. Eppinger 2001).Their power has been verified through a lot of real world applications from various fields, including automotive (S. D. Eppinger et al. 1994; McCord et al. 1993; Yassine 2006), aerospace (Avnet & Weigel 2010; Browning 2001; Jameson 1999), construction (Austin et al. 2000), or telecommunications (S. Eppinger 2001), to name only a few.

The choice of the type of DSM depends on the application, Browning argued that they could be used to (i) represent a design architecture based on components and/or subsystems (component-based), (ii) model process and activity networks based on activities and their information flow and other dependencies (activity-based), (iii) model organization

structures based on people and/or groups and their interactions (team-based), or (iv) model low-level relationships between design decisions and parameters, systems of equations or subroutine parameter exchanges (parameter-based).

Browning identified a series of methods based on DSMs that can help enhancing the performance of the design process: (i) partitioning or sequencing (i.e. reordering) to reduce the number of feedback loops in the dependencies and therefore increase the performance of the design process, (ii) tearing to choose the set of feedback marks that, if removed from the matrix, will render the matrix upper-triangular, (iii) banding to show independent (i.e. parallel or concurrent) activities (or system elements), (iv) clustering to find subsets of elements (i.e. clusters or modules) that are mutually exclusive or minimally interacting subsets. The type of analysis depends on the goal of the DSM (Table 2).

Table 2: Relevant analysis methods for the different types of DSMs (Browning 2001)

	Representation	Application	Analysis method
Task-based	Task/Activity input/output relationships	Project scheduling, activity sequencing, cycle time reduction	Partitioning, Tearing, Banding
Parameter-based	Parameter decision points and necessary precedents	Low level activity sequencing and process construction	Partitioning, Tearing, Banding
Team-based	Multi-team interface characteristics	Organizational design, interface management, team integration	Clustering
Component-based	Multi-component relationships	System architecting, engineering and design	Clustering

DSMs have even been introduced in literature to other extended applications through methods such as:

- using numeric entries instead of binary ones (S. D. Eppinger et al. 1994). Since the binary DSM represents only strict precedence relations and does not provide a metric on its quality, in complex design these analysis are usually crowded with weak dependences, and this leads to an extremely useless coupled design matrix. The numeric entries can represent a measure on issues such as the degree of dependence, task duration (if the DSM is task-based), electrical or vibrational characteristics, parameter sensitivity, historical variance of task results, certainty of planning estimates, or volume of information transfer.
- using different levels of abstraction by introducing a distinction between intra-domain DSMs that represent the detailed interactions inside a specific subsystem,

and inter-domain matrices, designated as Domain Mapping Matrices (DMM) to represent interdependencies (Danilovic & Browning 2007).

2.5. Computer Supported Collaborative Design (CSCD)

CSCD is becoming a promising field that focuses on the application of collaborative efforts into the design phase. Its depth and breadth is far beyond the traditional definition of CE (Hartley 1992), it is not specifically tailored for the conceptual design phase and deals with systems developed not only among teams within the same company, but also across the boundaries of companies and time zones. Yet, since the type of interactions at the conceptual stage are usually similar to the ones on later design phases, most of the information and communication technologies that are used to augment the capabilities of the individual specialists in CSCD can also enhance the perception of complex engineering systems during the conceptual design phase. The commonalities between CSCD and the objectives of this work justify a special consideration to the past and current applications related to this field.

CSCD as a field started when the human-machine evolved towards research human–human interaction via networked computers in the 1980s. It is in the 1990s that engineering design is recognized as a multidisciplinary field and CAD/CAM/CAE (Computer-Aided Design/Computer-Aided Manufacturing/Computer-Aided Engineering) tools become standard practice. Although these are usually very good at representing the final product with great detail, they are poor in providing the rationale and design process evolution, and therefore do not address the needs of multidisciplinary design (Tomiyama 2006).

The last 20 years brought a high number of new CSCD contributions dealing with both synchronous and asynchronous collaboration modes, mainly on:

- groupware used to facilitate communication among engineers and designers, from simple emails to group calendars, or videoconferencing applications,
- software agents used to integrate different engineering tools. Examples of early applications of software agents in collaborative design include PACT (Cutkosky et al. 1993), DIDE (Shen & Barther 1995), and SINE (Brown et al. 1995), and,
- more recently, web services, and computing grids for collaborative design (Shen et al. 2008).

Although most of the early studies focused on asynchronous collaboration for its convenience to not fix designers to specific timeframes, some of the interactions inside a design team are not easily captured and recorded in this way. Designers might feel it is impersonal and they prefer a synchronous approach (W. D. Li & Qiu 2006). Despite the additional cost and logistical requirements to «get everyone together at the same time», synchronous collaboration has the potential to meet collaborative requirements more efficiently in an agile environment like a CE environment for the conceptual design of complex engineering systems. A correct balance between synchronous and asynchronous collaboration is critical to maximize the performance of the design process and simultaneously provide room to creativity like in the standard spiral process.

Relevant work made in this field supports that the main goal of CSCD is to address key technological requirements to develop technologies to support (Molina et al. 1995; Reddy et al. 1993; Shen et al. 2008; Kahaner & Lu 1993):

- *computer-aided collaborative decision support* to enable groups and teams to interact, collaborate and make decisions during the design process (§2.5.1).
- *collaborative design tools* that organize the SE system development process itself and serve as reference models to establish the time-ordered application of people, methods and tools during the design process (§2.5.2).

These support technologies are built on top of distributed computer-based information architectures that are not the main focus of this work. These are the standards and hardware infrastructures required to achieve the high levels of integration and communication necessary during the design phase. The Standard for the Exchange of ProductModelData (STEP) (steptools.com n.d.) is an ISO standard aiming at a complete unambiguous, computer-interpretable definition of the physical and functional characteristics of a product throughout its lifecycle using technologies such as the eXtensible Mark-up Language (XML) (w3.org n.d.) or the Virtual Reality Mark-up Language (VRML) (Omar et al. 2009) to enable better cross-platform and cross-enterprise exchange of multi-media information and design models.

2.5.1. Computer aided collaborative decision support

The Moulton small-wheeled bicycle (1962) is the classic example mentioned in literature of a compromised decision during conceptual design (French 1985). Moulton began by questioning the need for large wheels on a bicycle since these were less compact, less stiff

and produced more air drag than small wheels. On the other hand, if small wheels had the same tyre section as large ones they would have to be inflated to a much higher pressure because of the short contact patch on the road, and they would then be much more uncomfortable. After evaluating the pros and cons, Moulton decided that his bicycle should be small-wheeled, of low resistance and comfortable, and the only way that could be done was by using narrow, high-pressure tyres and a separate suspension (Hadland 2000).

Computer aided collaborative decision support methods assist the designers engaged in decision-making activities with finding a solution that best suits their goal and their understanding of the problem. It provides a way to structure a decision problem, represent and quantify its elements, and evaluate alternative solutions. Computer aided collaborative decision support can be applied to (Forman. & Gass 2001):

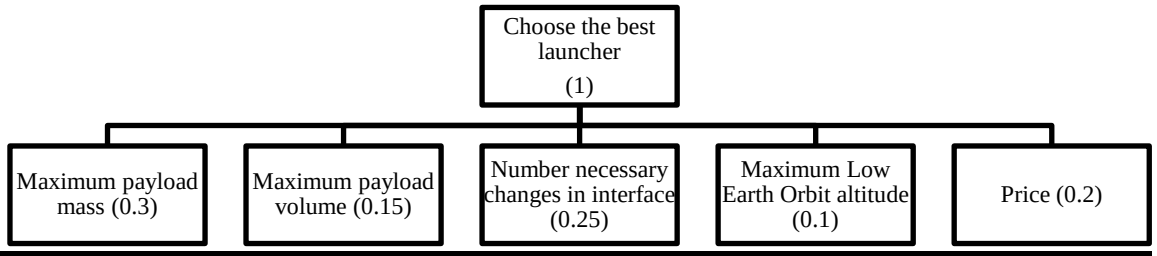
- choice: the selection of one alternative from a given set of alternatives, usually where there are multiple decision criteria involved,
- ranking: putting a set of alternatives in order from most to least desirable.
- prioritization: determining the relative merit of members of a set of alternatives, as opposed to selecting a single one or merely ranking them,
- resource allocation: apportioning resources among a set of alternatives,
- benchmarking: comparing the processes in one's own organization with those of other best-of-breed organizations,
- quality management: dealing with the multidimensional aspects of quality and quality improvement, and
- conflict resolution: settling disputes between parties with seemingly incompatible goals or positions.

These decision support methods are associated, in literature, with the fields of Multi-criteria Decision Analysis (MCDA) or Multi-objective decision analysis (MODA) that consider multiple criteria (or objectives) in complex decision-making environments. The following sections introduce three types of methods approached by this work to support the methodology proposed in Chapter 4. Some of the tools developed to support decisions in a CE environment are introduced in section 2.5.2.

2.5.1.1. Analytic Hierarchy Process (AHP)

The roots of computer supported collaborative support can be traced to the work of Thomas L. Saaty in the 1970s, who focused on a method for organizing and analyzing

complex decisions called the: *Analytic Hierarchy Process* (AHP). This method results from an application of mathematical and psychological approaches to group decision-making (Saaty 2008) and is still used around the world in government, business, industry, or healthcare applications.



	Maximum payload mass (kg)	Rank	Maximum payload volume (m^3)	Rank	Number necessary changes in interface	Rank
Soyuz	15000	0.5	12	0.00	2	1.0
Proton	17000	0.7	23	0.85	3	0.5
Ariane 5	20000	1.0	25	1.00	4	0.0
SeaLaunch	10000	0.0	18	0.46	3	0.5
Weight	**0.3**		**0,15**		**0,25**	
Max value	20000		25		4	
Min value	10000		12		2	

	Maximum Low Earth Orbit altitude (km)	Rank	Price (MUSD)	Rank	Total Result
Soyuz	600	0.00	350	0.2	**0.44**
Proton	1200	0.67	250	0.6	**0.65**
Ariane 5	1500	1.00	400	0.0	**0.55**
SeaLaunch	1000	0.44	150	1.0	**0.44**
Weight	**0.1**		**0.2**		
Max value	1500		400		
Min value	600		150		

Figure 16: Example AHP applied to the choice of a space launcher (all values are merely descriptive)

The AHP is particularly tailored for complex problems involving subjective perceptions and starts by decomposing the decision problem into a hierarchy of tangible and measurable parameters which are then weighted by direct comparison (two at a time) to establish a measure of impact. The AHP uses those weights to evaluate the different options that have to be considered during the decision process (Figure 16). The applications of the AHP to complex decision support have grown exponentially and produced extensive results in problems involving planning, resource allocation, priority setting and forecasting. The AHP is part of the total quality management philosophy and is the basis of the Balanced Scorecard.

2.5.1.2. Multi Attribute Utility Analysis (MAUA)

The Multi Attribute Utility Analysis (MAUA) is a technique used to support decision makers who are faced with numerous and conflicting evaluations. This analysis provides

information on the impact of the different attributes of a system on the decision of the designer.

MAUA has emerged as a powerful tool for material selection in 1994, but its capabilities are nowadays used in a large array of problems from operations research to marketing. It is especially targeted to deal with complex measures that do not usually exhibit analytical models, such as performance, or manufacturability, that can be decomposed into a combination of several other measurable attributes such as price, weight or time (Roth et al. 1994).

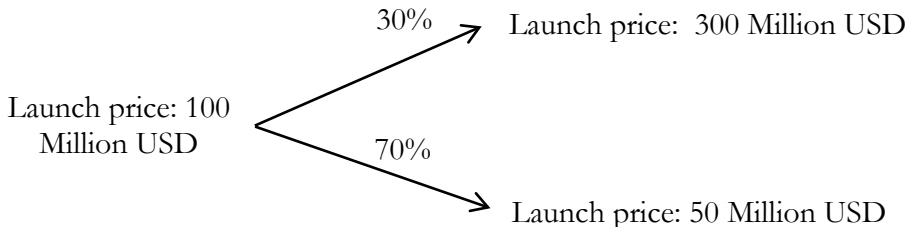


Figure 17: Example result lottery questions for MAUA. Looking at the price attribute, the designer is indifferent between having a launch system with a price of 100 Million USD or a different system that can cost 50 Million USD with a probability of 70%

Utility theories map preferences for an attribute into a normalized value known as utility (value between 0 and 1) by combining single-attribute utility functions into a single function that quantifies how a decision maker values different attributes relative to one another, taking into account the levels of each attribute (Ross et al. 2004). The different single utility functions are found through a series of lottery questions designed to find the point at which the designer is indifferent between having a definite option or an option with a certain probability (e.g. Figure 17).

The value of the multi attribute utility function is simply a weighted combination of the single attribute functions, where the weight is found by exploring the boundaries of the design interval (e.g. when finding the multi attribute utility function to combine launching price and payload mass, the weight of the launch price attribute can be found by determining the indifferent point between a certain system with minimum price and minimum payload mass, and another option with the minimum price and the maximum payload mass with a certain probability). The designers can then base their decision on the utility distribution of the different design possibilities (Figure 18).

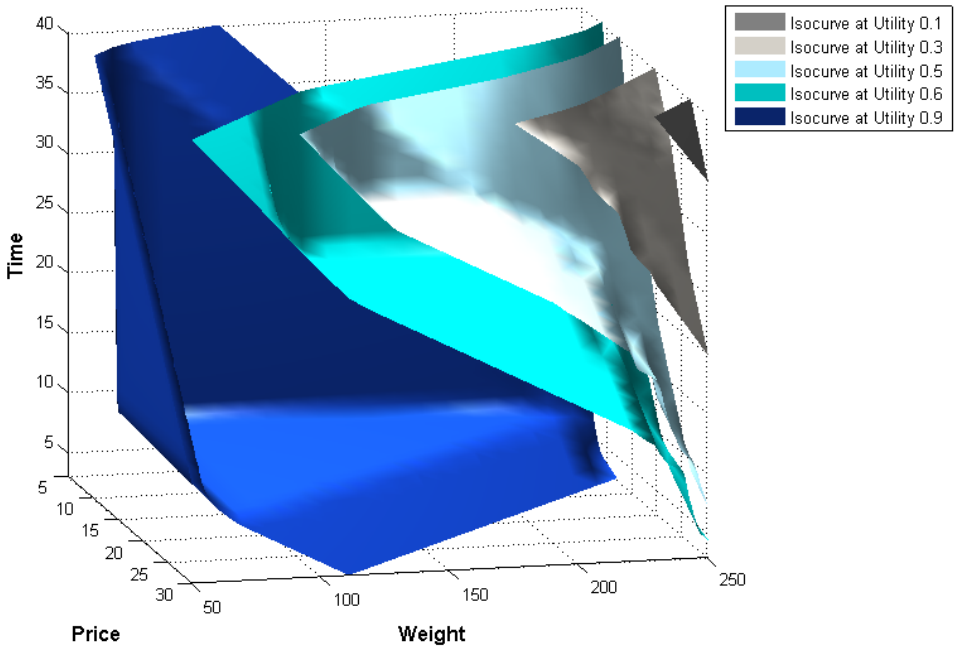


Figure 18: Example of multi attribute utility function combining weight, price and time

Using MAUA to analyze judgmental complex decisions during the conceptual design phase can yield significant results in the quality of the final system and simultaneously establish a clear protocol explaining the reason behind a specific choice that was taken.

2.5.1.3. Agent-based decision support

Agent-based modeling started when the Von Neumann architecture arrived to the IT field in the late 1940s, but its general use did not arrive until the 1990s, when the IT capabilities matched the requirements of the necessary applications. It simulates the actions and interactions of autonomous agents in order to assess their effects on the system as a whole. The idea behind agent-based modeling is to simulate complex behaviors by using simple models of agents that can be real or virtual, and defining rules on how agents feel the environment they are integrated in, how they behave, and how they communicate with the others (Axelrod 1997). Examples of applications include optimization, modeling consumer behavior, social network effects and workforce management.

When dealing with collaborative environments, agent-based modeling is mostly used to support cooperation among designers, enhance interoperability between computational tools, or allow better simulations. A more detailed review of several well-known applications can be found in (Shen 2001).

The agent characteristics, like autonomy, ability to perceive the surrounding environment, act in specialized domains, and their capability to cooperate with other agents, made it

useful in applications like decision support by using a loosely coupled network of problem solvers that work together to solve problems that are beyond their individual capabilities (Shen et al. 2008).

Although it is a quite powerful approach to deal with deterministic interactions, it proved itself to be insufficient to support dynamic collaborative design environments, where the decisions usually involve complex and/or non-deterministic interactions usually arising from the inclusion of judgmental perception resulting from human experience (Shen 2001). A successful implementation of these methods in a CE environment would require restraining its use to problems with deterministic behaviors.

2.5.2. Collaborative design tools

Collaborative design tools are used to support the different activities throughout the life cycle of a product, from conception to maintenance. They can be used in applications such as (French 1985): increasing insight into problems, and the speed of acquiring insight; diversifying the approach to problems; reducing the size of the mental step required in the design process; prompting inventive steps, and reducing the chances of overlooking them; or generating design philosophies (synthesizing principles, design rationales) for the particular problem in question. These tools are part of the basic infrastructure requirements for conducting CE design and should support qualitative reasoning, quantitative evaluation and optimization.

Although they have been introduced in the 1980s, the low IT resources at the time delayed its general use until the 1990s. It was then that the first hardware and software architectures that enabled integration, execution, and communication among diverse disciplinary processes arrived. However, in 1999, Rodgers expressed that "an examination of current frameworks reveals weaknesses in various areas, such as sequencing, displaying, monitoring, and controlling the design process" (J. L. Rodgers et al. 1999).

Since then there have been numerous efforts to increase the performance of these tools in collaborative environments such as CE. The following sections detail some of the tools analyzed in the scope of this work focusing on:

- *design process monitoring and optimization* tools, to increase the designer's perception of the global status of the design process and guide on possible changes to increase its performance (in terms of design process duration) (detailed in §2.5.2.1),

- *design selection* tools, to support the decisions during the conceptual design phase based on the methods introduced in the section 2.5.1 (detailed in §2.5.2.2),
- *design optimization* tools, focusing on the optimization of the design solution at the conceptual phase (detailed in §2.5.2.3),
- *integrated collaborative design* tools, focusing on system architectures that support the complete conceptual design by integrating all the other decision support tools in a common environment (detailed in §2.5.2.4).

2.5.2.1. Design process monitoring and optimization

Design process monitoring and optimization tools have gathered much attention from companies, organizations and universities, most notably in the aerospace industry (Ma et al. 2008). Monitoring the design process in order to optimize its performance is especially important in a CE environment to facilitate a more structured approach to the highly iterative process of conceptual design of complex engineering systems (Avnet & Weigel 2010).

One of the most prominent tools for this purpose is the Design Manager's Aid for Intelligent Decomposition (DeMAID). It was developed at the MIT in 1999 (J. Rodgers 1999) and, since then, it has considerably evolved until a more robust, designer-centered tool. It consists of a knowledge-based software tool for minimizing the feedback couplings; sequencing the design processes; grouping processes into iterative sub cycles; decomposing these sub cycles into a hierarchical, multilevel structure for a design cycle; and displaying the sequence of processes in a DSM format. It uses a genetic algorithm to rapidly examine many possible sequence combinations (J. Rodgers 1999).

DeMAID is essentially an asynchronous tool where the data insertion is completely detached from the design process itself. The designers insert information about the status of the design process after each design loop and expect that to infer possible improvements for the next iteration. Since it requires a lot of work and time from the designers, it is usually accomplished by a dedicated person who delivers asynchronous reports on the overall status of the design (Avnet & Weigel 2010). In a CE environment, the speed at which the information is made available is very important so, since the results of DeMAID often take a considerable amount of time, they are not considered until the next similar design.

A more recent tool, the VDSM (Visual Design Structure Matrix) is aiming at a more synchronous approach to monitor the design process. It provides insight into individual aspects of complex design problems by means of a DSM representation of the problem using a web-based interface (Agrawal et al. 2004). Although it is still not real-time, current research efforts are focusing on integrating its capabilities with a geometric view of the different parts (Kenneth W. English & Bloebaum 2008) coupled with cost and error models, enabling designers to explore the possibility of eliminating or suspending couplings interactively, before or during a design iteration.

2.5.2.2. Design selection

Design selection tools are used to guide the designers from an initial space of possible solutions to the final design solution. Early research work in this field focused on deterministic selection, which was the way most American car companies went about design. Deterministic selection refers to the lack of variability, considering that point solutions give an accurate representation of what is expected in reality within the same order of magnitude. This is a restrictive assumption and is applicable only for some special cases (Keeney 1976) since it makes the assumption the final design will be nothing more than a more detailed version of earlier point design estimations.

In order to comply with preference variability, caused for example due to lack of information on customers' needs or technological constraints, other tools started to focus on providing a range of preferences to enable the choice of «potentially optimal designs» for that range. This approach is part of a lean manufacturing strategy introduced by Toyota whose goal is to consider a broader range of possible designs and delay certain decisions until later design phases. This large group of ideas is eventually narrowed down to the final solution which is usually the best solution of the set. The authors that propose tools allowing preference variability advocate that their use in a CE environment allows a robust solution to be found without much need for optimization (Womack 1991).

Sensitivity analysis is one of the most prominent methods to study how the variation in the output of the system can be attributed to different variations in the inputs, to find the degree of "robustness" of the preferred design(s) to preference variation (Morse et al. 2006). It can be completed with simple worksheets or other calculus tools, but the usual approach in sensitivity analysis is to change one factor at a time to increase the comparability of the results, which can disguise dependencies amongst the different parameters (Saltelli & Annoni 2010).

Recent research has focused on developing tools for design tradespace exploration to approach the preference variability of several parameters at a time and also include attribute variability due to uncontrollable behavior of the system such as manufacturing errors and price changes due to market crisis.

One of these tools, MATE-CON (Ross et al. 2004), was born from the realization that system requirements do not adequately consider the full range of possible designs and their associated costs and utilities throughout the development and lifecycle, which can lead to long design times and designs that are locally optimized but may not be globally optimized. Instead of a single solution, there should be an exploration of unconstrained dimensions to allow decision makers to analyze interactions of design variables. MATE-CON was developed at the MIT and is a multi-attribute tradespace exploration tools especially targeted at CE environments with the goal of generating and evaluating a multitude of system designs simultaneously and establishing a common metric in the form of a multi-attribute utility function (§2.5.1.2) to create a common metric for the evaluation of the different design possibilities. MATE-CON does not support the fact that «everything can be reduced to a number and equation». This is seen clearly in the employment of CE that maintains the fundamental need for the «human in the loop» (Diller 2002).

Another similar tool using a Java visualization support was developed by the Applied Research Laboratory (ARL) at The Pennsylvania State University and is called ARL Trade Space Visualizer (ATSV) (M. Yukish et al. 2007).

The automation used in these tradespace exploration tools enables the analysis of a large number of designs in a short period of time. Despite these benefits, these tools still need to be matured in some areas: faster graphical interfaces need to be developed; new visualizations need to be used; and the user needs to understand the impact of problem size and complexity on performance to adapt the depth of the analysis to the size of the problem (M. Yukish et al. 2007).

2.5.2.3. Design optimization

When talking about design optimization tools for complex engineering systems, we are usually talking about Multidisciplinary Design Optimization (MDO), the field of engineering that uses optimization methods to deal with design problems involving different subsystems with mutually interacting phenomena. The performance of complex engineering systems depends on the interaction of many disciplines whose behavior is

governed by a very large set of coupled equations. In practice, engineers deal with these equations by partitioning them into subsets corresponding to the major disciplines. In this process of pragmatic partitioning, the couplings among the subsets tend to be reduced in number because it is burdensome to strictly account for them all. Couplings are retained or neglected on the basis of what is known or assumed about their strength in a particular vehicle category. Generally speaking, the more complex the system is, the more couplings should be accounted for.

The evolution and diversification of optimization algorithms along with the development of support technology, especially in terms of computer calculation capabilities, led to an escalating number of optimization algorithms such as: Gradient, Newton, Quasi-Newton, Non-linear least-squares and Robust programming. An exhaustive list of these algorithms is far beyond the goal of this work, the reader may refer to (Pedregal 2003) for additional details.

In addition to the optimization algorithms, the formulation of the MDO problems has also progressed towards a multitude of possibilities such as: Multiple-discipline-feasible (MDF) when the overall problem is considered in its total complexity involving expensive calculations, or Individual-discipline-feasible (IDF) when a set of surrogate variables is used to decouple the overall problem into smaller sized subsystem level ones. A detailed explanation of all the MDO formulations can be found in (Zopounidis 2010).

Research in MDO has resulted in a wide variety of tools and techniques to assist in the analysis and optimization of these highly complex systems. Most of them are commercial tools (modefrontier.com n.d.; iosotech.com n.d.; redcedartech.com n.d.), but there are also open source collaborative efforts in this field such as the DAKOTA project (dakota.sandia.gov n.d.). The ever growing availability of IT resources allows products to be better analyzed earlier in the design process. More accurate models can support the designers in exploiting trade-off opportunities earlier in the design process. However, some authors mention that the existing tools still suffer from data integration problems for collaborative and platform-independent MDO implementations (Agrawal et al. 2004).

Choosing an MDO approach can lead to a more or less automatic method where design selection is usually kept inside a «black box» with the human outside the loop. A successful implementation of design selection and optimization strategies in a CE environment will

depend on the correct balance between the use of MDO methods for design models with little variability and tradespace exploration strategies for higher-level analysis.

2.5.2.4. Integrated collaborative design tools

Most of the precursor work on balancing synchronous and asynchronous CSCD systems focused on dedicated solutions for specific problems or specific tasks such as decision support. Unfortunately, when dealing with complex systems, there are often different strategies inside the same design team, which leads to the need of establishing an integrated environment to define and develop information models, integrate and implement decision support applications and provide adequate information system architectures.

This need was clearly identified as early as 1995, by Molina, who believed that integrated collaborative design tools for CE should ideally (Molina et al. 1995):

- capture and represent product information, and manufacturing process and resource information;
- provide immediate access to information about previous product or process design and transmit design information without loss of intent or detail;
- offer immediate access to information about manufacturability, reliability, maintainability, safety, performance, and other elements of the life cycle;
- allow access to the most current state of the product or process configuration description as it is being developed;
- keep data to be shared by team members in commonly accessible data bases.

There were several early attempts to establish these integrated environment system architectures that supported the two obvious domains of cooperation: space (same or different) and time (synchronous or asynchronous), using a repository of information and providing a means by which concurrency in design may be achieved. One of the them, the Synchronous/Asynchronous Common Environment - Computer Supported Cooperative Work (SACE-CSCW) (dos Santos et al. 1997), allowed a definition of different dimensions such as group/subgroup, user roles, public or closed type of meeting, anonymous or nominated method of meeting and three modes of cooperation: full view, partial view and single view. It also included brainstorming, organization/consolidation, voting, and action sessions of a typical decision-making meeting. Unfortunately its interoperability constraints required specific training, demanding a lot of time and effort from the designers.

In order to approach these interoperability problems, some tools followed a standalone approach where the definition of the different subsystem models was fixed for specific applications such as space systems (Starnone 2005). Although they proved themselves to be very efficient in solving particular problems, their rigidity limited their applicability to incremental design solutions (no way to include revolutionary ideas).

Some of the recent CSCD applications that focus on Product Data Management (PDM) and Product Lifecycle Management (PLM) have had more success in terms of market acceptability. Mainstream solutions that are now being used by industry to facilitate engineering design include: UGS TeamCenter (Teamcenter 2011), PTC Windchill(ptc.com 2011), ENOVIA VPLM, ENOVIA MatrixOne and ENOVIA SmarTeam (ENOVIA 2011). These systems focus on the overall system's lifecycle and can provide: workflow and process management, design change management, and VM-based collaborative workspaces to retain the visualization information of product models and allow manipulation of the product models, such as rotation, measurement, annotation, and mark-up (Shen et al. 2008).

In PDM/PLM systems the design process is considered an integral part of a global lifecycle analysis. There are numerous successful case studies showing the application of these systems during the detailed design phase but, when dealing with conceptual design of complex engineering systems, PDM/PLM tools still suffer from lack of flexibility, insufficient openness, and lack of usability to both end-users and systems maintainers (Shen et al. 2008). During the conceptual design phase, the different subsystem models are often not very detailed (no CAD files for example), restraining these models at an early phase only to use PDM/PLM systems is not advisable and can yield considerable performance losses in the final design solution.

The recent Open Concurrent Design Server (OCDS) is an initiative of the European Space Agency (ESA) that aims to overcome the limitations of PDM/PLM during the conceptual design phase (Richardson & Dunne 2010). Its goal is to provide the building blocks of a CE environment for the European Space Industry, using Open Standards Information Models and Reference Libraries. At the same time it distributes an open data exchange standard for early phase space system engineering and design activities described in the standard ECSS-E-TM-10-25 (ECSS-TM-E-10-25A 2010). The OCDS provides a software framework to tackle problems specific to the conceptual design phase by: increasing data management capabilities, achieving a hardware and software independent solution,

including the possibility of optimizing the design process and streamlining the communication between the different actors that take part in a CE environment. Unfortunately, although this project started in the beginning of 2009, its technical complexity undermined initial results which only now begin to arrive.

Solutions like the OCDS seem to answer the need for integrated environments at the conceptual design phase in the near future. However, since they are based on specific technologies which require a reasonable amount of learning commitment from the designers, there seems to be a gap between the system engineers (who perceive this as a very important tool) and subsystem designers (that do not seem to grasp the advantages to their specific work). It is important to focus future research work on integrating tools commonly used by subsystem designers (e.g. spreadsheets) to create shorter learning curves and, therefore, increase the adoption amongst all the members of the design team.

2.6. Major issues revealed in literature

This chapter focuses on detailing the precursor work that was on the onset of this work by providing a state of the art of the fields adjacent to the main theme of the work, as well as their shortcomings that need to be addressed during the upcoming years. The following table provides an overview of the main issues identified and detailed earlier.

Table 3: List of main issues revealed in literature

Field	Issue
Design processes (detailed in §2.1)	When dealing with a CE environment there is a need to switch from a linear waterfall process to a spiral process. Yet, the lack of guidelines in a pure spiral approach can guide to performance losses.
	The design process activities may extend over potentially long period of time, well over the conceptual design phase. The context of design must be maintained over that period and longer, to address life-cycle issues and to allow for future reuse (Reich et al. 1999).
	Each member of the team should be aware of the design process status at all times, lending its expertise to develop the best possible product within given cost and schedule constraints.
VM (detailed in §2.2)	It is necessary to review the possible VM and identify (i) the most relevant for complex engineering systems, capable of doing all the desired operations, and (ii) the VM that are equivalent, to avoid unnecessary repetitions and implementation effort.

VM (detailed in §2.2) (cont.)	Designers with the same expertise or from the same discipline may have different perspectives about a particular system behavior. VM should allow changing these perspectives as well as letting them evolve in response to the context of a particular project.
	Designer-induced and cognitive user-induced VM pitfalls need to be reduced for a successful implementation in a CE environment.
	Designers need to interact with the system and negotiate with peers via VM. The challenge is to make intelligent interfaces available to all resources so that the designers will have more flexibility to conduct efficient and effective design. The interfaces should be integrated, expressive, goal oriented, cooperative, easy to use, and customizable (Shen et al. 2008).
	VM should allow the designers to get a quick general perspective of the system, displaying how the different components of the systems, parts and subsystems interact.
	VM should stimulate creativity and therefore need to be dynamic, iterative. They should guide on possible viewpoints that can be used by the different designers but still leave some room for the designer's cognitive tasks. They should accept that at the conceptual stage the design models are not mature enough to use only one type of very complex viewpoint and instead guide the design process to develop these same design models.
KM (detailed in §2.3)	It is necessary to convince both engineers and management of the cultural, technical and organizational requirements to build a usable and useful KM strategy. It is important to emphasize meritocracy, transparency.
	Analyzing the knowledge of individual subsystems independently does not yield good results with complex engineering systems, since they cannot be reduced to a sum of simpler parts that can easily be quantified and independently analyzed. KM should be done in an integrated environment accessible to everyone.
	The design team often uses different languages or terminologies to describe disciplinary knowledge. KM should enable the capture of unstructured/informal pieces of information such as text, images, audio and video; as well as structured/formal, such as equations and 3D models.
	Individual designers in a CE environment may come from different cultures (e.g., egalitarian or capitalistic) whose impact on the adoption of KM technologies may be significant. A successful KM implementation should focus on the openness towards contributions from everyone.
	KM tasks should be as automated as possible to reduce the designer's workload. They need to be intuitive and accessible at all times.
Design modeling (detailed in §2.4)	An abundance of modeling languages can pose a barrier for adoption from the industry's point of view.
	Some of the most prominent languages used nowadays in MBSE are complex and contextualized (language depends on the type of diagram), and require a reasonable amount of effort from the designers' point of view.

Design modeling (detailed in §2.4) (cont.)	DSMs can be useful designing modeling methods but they are still asynchronous methods whose major findings cannot be used until the next similar design. Future work needs to focus on turning these methods into a synchronous collaborative approach.
CSCD (detailed in §2.5)	A successful implementation of CSCD needs: a series of new strategies, including an efficient communication strategy for a multidisciplinary group of people from the design and manufacturing departments to share and exchange ideas and comments; an integration strategy to link heterogeneous software tools in product design, analysis, simulation and manufacturing optimization to realize obstacle-free engineering information exchange and sharing; and, an interoperability strategy to manipulate downstream manufacturing applications as services to enable designers to evaluate manufacturability as early as possible (W. Li 2006).
	CSCD agent-based decision support methods are a quite powerful approach when dealing with deterministic interactions but proved themselves to be insufficient to support dynamic collaborative design environments, where the decisions usually involve complex and non-deterministic interactions, and judgmental perception (result of human experience)(Shen 2001). A successful implementation of these methods in a CE environment would require restraining its use to the applications with deterministic behavior and using other methods such as AHP and MAUA for non-deterministic behaviors.
	Most of CSCD tools described in the literature are developed for the collaboration of experts familiar with the use of complex IT tools, it is important to develop new tools with shorter learning curves avoiding the complex issues that experts have to face.
	CSCD tools should support the system's evolution, dynamism, and history management (Reich et al. 1999).
	Design teams communicating via a CSCD tool perceive an increase in mental workload and interact less frequently, but for a greater total amount of time (Hammond et al. 2005).
	Design process monitoring tools should focus on becoming completely integrated with the design process and avoid separate asynchronous actions.
	Design selection tools focusing on a tradespace exploration paradigm still need to be matured in some areas: faster graphical interfaces need to be developed; new visualizations need to be used; and the user needs to understand the impact of problem size and complexity on performance to adapt the depth of the analysis to the size of the problem (M. Yukish et al. 2007).

CSCD (detailed in §2.5) (cont.)	Some of the recent MDO tools still suffer from data integration problems for collaborative and platform-independent implementations (Agrawal et al. 2004).
	A successful implementation of design selection and optimization strategies in a CE environment will depend on the correct balance between the use of MDO methods for design models with little variability, and tradespace exploration strategies for higher level models.
	CSCD tools are evolving towards fully integrated tools connected through the network covering the full product lifecycle from conceptual design, to detailed design (with detailed modeling, simulation and optimization), virtual prototyping, manufacturing, service and maintenance, in standard PDM/PLM packages. However these tools still suffer from lack of flexibility, insufficient openness, and lack of usability to both end-users and systems maintainers (Shen et al. 2008). During the conceptual design phase the different subsystem models are often not very detailed (no CAD files for instance), restraining these models at an early phase to only use PDM/PLM systems is not advisable and can yield considerable performance losses in the final design solution.
	Current CSCD tools are based on specific technologies which require a reasonable learning commitment from the designers, there seems to be a gap between the system engineers (who perceive this as a very important tool) and subsystem designers (that do not seem to grasp the advantages to their specific work). It is important to focus future research work on integrating tools commonly used by subsystem designers (e.g. spreadsheets) to create shorter learning curves and, therefore, increase the adoption amongst all the members of the design team.

[*This page intentionally left blank*]

3. COLLABORATIVE DESIGN ISSUES

The previous chapter introduced a series of issues found in the literature regarding design processes, visualization of complex engineering systems, KM, design modeling methods, and CSCD. This chapter focuses on describing the research strategy adopted to approach these shortcomings, and introduces the general ideas that led to the main contributions of this work.

The formulation of the contributions of this work started with a survey made to designers with experience in a CE environment. This survey consisted on a set of open and closed questions planned to gain insight into the main problems felt by the different subsystem designers, and obtain a quantitative measure on their level of satisfaction.

Combining the issues identified in literature with the feedback of the survey led to a set of preliminary ideas to tackle the current ubiquitous problems in conceptual design. These ideas, detailed in section 3.2, represented the backbone of the main contributions of this work: an integrated conceptual design methodology (detailed in Chapter 3) and two enabling tools: a decision support tool (detailed in Chapter 5) and an INtegrated Concurrent Real-time EnvironMENT (INCREMENT) (detailed in Chapter 6).

3.1. Survey to designers in a CE environment

A survey was performed in May 2009 to the designers of the Concurrent Design Facility (CDF) at the European Space Research and Technology Centre (ESTEC) of the ESA. The survey consisted on a total of 26 questions (8 open and 18 closed) and had a total of 18 valid answers from designers of complex space missions, which are good examples of complex systems (the structure of the survey is detailed in Appendix 1). The characteristics of the sample distribution are detailed in Figure 19, they were mostly men (94%), with at least a Masters education (100%), between 20 and 40 years old (95%), and with more than 4 studies of experience in a CE environment (72%) (one study corresponds to the complete conceptual design phase of a space system).

Their fields of expertise were broad and covered most of the subsystems found in a typical space system: Mission Analysis, Structures, Propulsion, Power, Risk Analysis, Cost

Analysis, Configuration, Simulation, Guidance Navigation and Control, Aerothermodynamics, Instruments/Optics, Communications, and Systems Engineering.

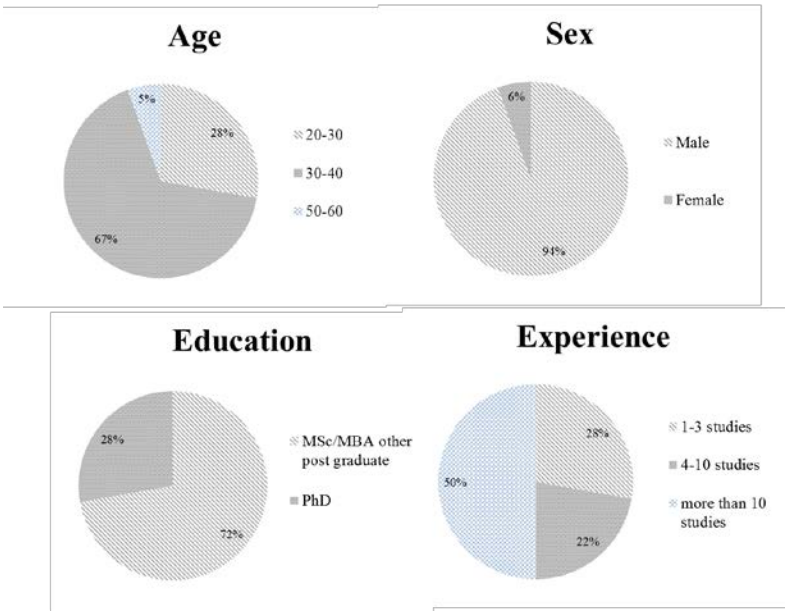


Figure 19: Sample distribution for the survey

A standard question was asked to understand if there was a convergence with respect to the perception of complexity from the designer's point of view. They were asked to rank a series of systems from 1 to 5 according to the expected complexity during the conceptual design phase (5 being an extremely complex system) (Figure 20). There was a general agreement with respect to the most complex systems, but there was a high dispersion in the results when evaluating the less complex ones.

The results exhibited a problem already identified in literature (§1.2), that there is no standard metric to determine the complexity of a system, making it a highly subjective evaluation task. There is a tendency to describe a system that is more dependent on one's field of expertise to be more complex than another system on which the direct contribution is smaller. This behavior was evident when analyzing the results. For instance, the propulsion specialist considered the design of a rocket nozzle to be much more complex than the rest of the specialists, and the telecommunications designers considered the antenna to be one of the most complex systems in the list. Since the concept of complexity is difficult to grasp, people have the tendency to associate it with technical difficulty so, when they are aware of the technical difficulty of a specific system, they tend to consider it as very complex, neglecting the other dimensions that can add to complexity

such as size of the system (number of subsystems and intricacies that have to be considered), or the socio-cultural factors (number of organization involved, etc.).

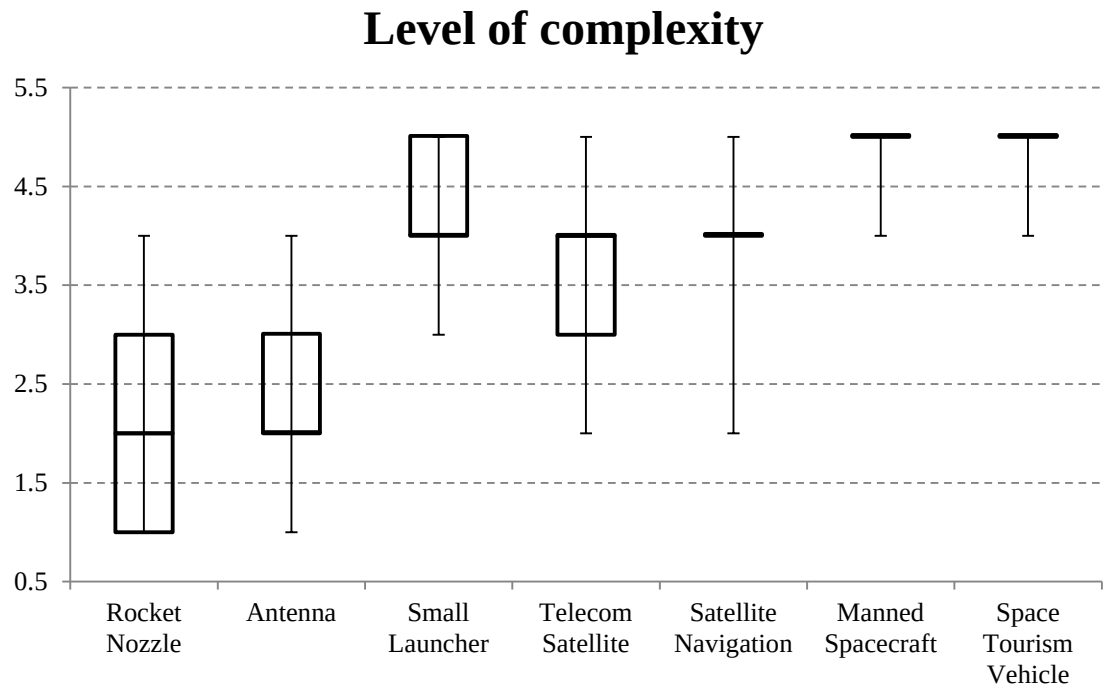


Figure 20: Survey results evaluating the level of complexity of different space systems. Box plot showing the distribution of the results (median, 25th and 75th quartiles, maximum, and minimum values) for a level of complexity ranging from 1 to 5 (5 being the highest level of complexity)

The other questions were more directed towards the use of CE at the conceptual stage, the designers were asked on their opinion about whether CE was the most efficient approach to design space systems at the conceptual stage, to which 83% of the respondents answered they either agreed or strongly agreed, the rest of the designers were neutral (Figure 21).

They were also asked about the overall impression on the current CE methodologies they had been using, 83% of the respondents showed their dissatisfaction by mentioning that these methodologies could or needed to be improved (Figure 21).

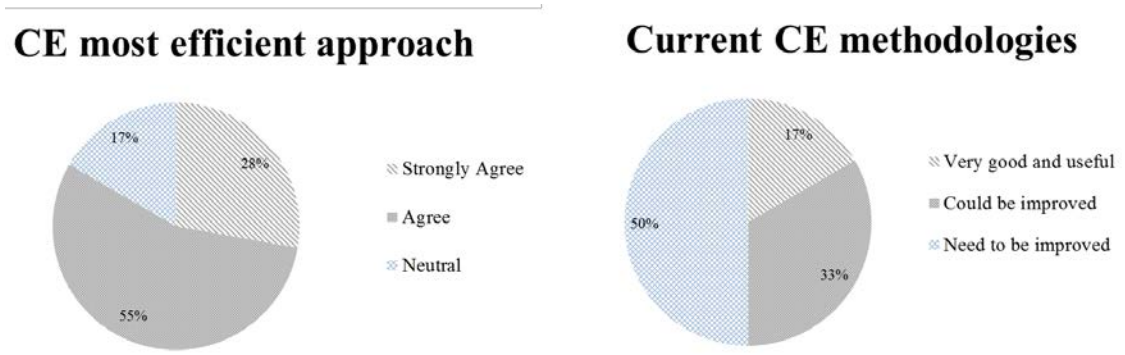


Figure 21: Survey results on the satisfaction about current CE approaches at the conceptual design phase

When the respondents were asked to rank from 1 to 5 (5 being the most important), the aspects on which existing CE methodologies or CSCD tools could be improved (Figure 22), despite the dispersion in the results, the representation was the one that exhibited the highest score (median 4.5) suggesting that designers feel the need to have an overall system perspective to which they can relate their subsystem choices. Integration was chosen as the second most important aspect, suggesting that the designers seek to integrate their subsystem level methodologies and tools in an integrated environment with high interoperability capabilities. The optimization, real-time update, and calculation time were considered as less important factors that, nevertheless, still need to be taken into account when building new tools or methodologies.

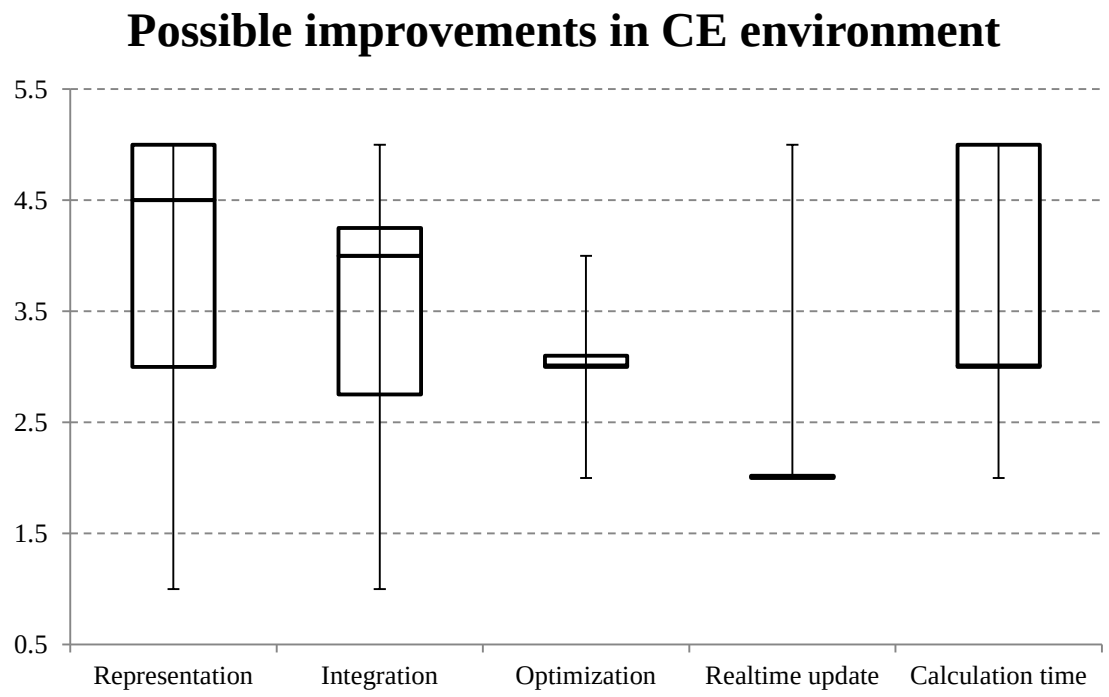


Figure 22: Survey results on the possible improvements for CE methodologies and tools. Box plot showing the distribution of the survey results (median, 25th and 75th quartiles, maximum, and minimum values) for a level of importance ranging from 1 to 5 (5 being the highest level of importance)

The designers were then asked to provide an idea of the main CSCD tools they had been using in a CE environment, as well as an idea of their advantages and disadvantages. Table 4 shows a breakdown to the subsystem level, on the type of tools and the overall comments made by the respondents.

Table 4: Survey results to the advantages and disadvantages of current CSCD tools used in a CE environment for the conceptual design of space systems

Subsystem	Tools	Good aspects	Bad aspects
Systems Engineering	Excel based tools for sharing data between the susystems in a concurrent environment	Easily accessible, easy to use	Do not allow a reasonable overview of the system, not robust, remarkably low level of automation
	MDO tools compatible with Matlab (mathworks.com n.d.), Excel and Java based models	Good performance tools	Do not allow a concurrent design approach
Mission Analysis	Ad hoc tools	Very detailed analysis	Standalone tools with no integration
	ASTOS (astos.de n.d.): optimization tool focusing not only the trajectory but also different system aspects of the vehicle: stage loading, propulsion properties, dry mass of tanks, etc. STK (Satellite Toolkit) (agi.com n.d.)	Good estimation of a preliminary design, which can be used as a starting design point within a CDF study	Interoperability, as the models for each discipline become more complex, the computational time increases a lot
	Matlab (mathworks.com n.d.) and EcosimPro (ecosimpro.com n.d.) for computation, simulation and visualization	Flexible enough to cope with most of the study	Lack of integration with the rest of the team, and consistency along different studies to increase the level of reusability and in-house resources for future studies
	Excel based tools	Simplicity	Lacks consistency, traceability, maintainability and integration among the different subsystems on top of the lack of power and reusability

Cost Analysis	Ad hoc tools ranging from equipment-level cost models (for the most detailed estimates) to models for launcher stages or spacecraft subsystems or even complete launchers and satellites.	Since they have been mostly developed in-house, they fit very well the needs	Very difficult to get accurate, detailed and up-to-date cost information. The motivation and participation of people varies largely, so you may end-up with partially up-to-date info and partially obsolete
Aerothermodynamics	CFD codes	Can be tailored, good support, quite powerful	Not user friendly, not easy to interface with each other
	Radiation simulation tools to support projects on (re)entry vehicles such as CHEMKIN (reactiondesign.com n.d.)		
	Engineering tools for quick assessment of orders of magnitude such as Matlab (mathworks.com n.d.)		
	Tools to calculate the trajectory of reentry vehicles		
	Excel based tools for sensitivity studies		
Risk Analysis	Item reliability software: reliability prediction, fault tree analysis	Powerful	Interoperability
	MS Access: risk identification, assessment and control.	Powerful	Interoperability
	Excel based tools: reliability prediction.	Easy to use, availability	Not persistent data, problems with interactions

Power	Mathcad (ptc.com n.d.) to develop mathematical models	Very powerful, flexible and user friendly	Difficult to integrate models from other subsystems
	PSIM (powersimtech.com n.d.) and PSpice (cadence.com n.d.) to simulate electrical circuits	Fast tools to develop or check electrical circuits, user friendly	Only useful for electrical circuits. difficult to integrate models from other subsystems
	EcosimPro (ecosimpro.com n.d.) to simulate a full power system	User friendly, very powerful, full visibility of the models, models from other subsystems can be easily integrated into a higher level model	Simulation data processing
	Excel based tools: information exchange with other domains	Enables the exchange of information with the other subsystems	Not user friendly, no visibility of the models implemented (in the end, the designer was not able to use the model for designing. Using his/her tools and then plugging the results in the Excel worksheets). Not reliable (many errors on the data exchange)
Communications	Software simulator (C++/Matlab) (mathworks.com n.d.) for the performance assessment of communication systems and subsystems	Accuracy and detail	Interoperability
	Excel sheet for link budget analysis		

Instruments/Optics	Zemax (zemax.com n.d.), a tool for optical design	Accuracy and detail	Interoperability
Propulsion	Excel based tools: propulsion system optimization within the context of the CDF.	Excel tools are generally easy to understand and have only a short learning curve before they can be used by newcomers	The current CDF tool is extremely manual in nature and required significant menial effort simply to populate with basic data. It does not allow for real interaction and feedback between the different sub-systems
	EcosimPro (ecosimpro.com n.d.) for propulsion system modeling	Sophisticated system modeling environment where interactions between sub-systems can be investigated.	Requires very careful definition of interfaces between subsystems, and careful management of heritage models for subsystems and components. The learning curve is longer, so it requires more from the designers
Guidance Navigation and Control	Matlab (mathworks.com n.d.), Mathcad (ptc.com n.d.), STK (Satellite Toolkit) (agi.com n.d.)	Detailed, powerful, flexible and user friendly	Interoperability
	Excel based tools	Simplicity	Rigid tool

These results agree with the main issues mentioned in section 2.6, showing that the different subsystems respondents recognize a lack of interoperability when using tools specific to the subsystem design, and exchange data with simple worksheets that, although offer an easy way to define mathematical models and being easily available, suffer from lack of user friendliness, visibility, traceability, and maintainability.

The survey went further on the representation issues and asked about two specific VM that could be implemented on enhanced CSCD tools: a view on the parameter dependencies showing how the inputs and outputs of the different subsystems related with each other, and a view of the impact of each design parameter on the downstream subsystem to provide a rough sensitivity analysis. The respondents answered that both the VM would be very beneficial during the conceptual design phase, 100% of the respondents agreed or strongly agreed to the use of a parameter dependencies' view, 94% answered the same way when asked about a subsystem impact view (Figure 23).

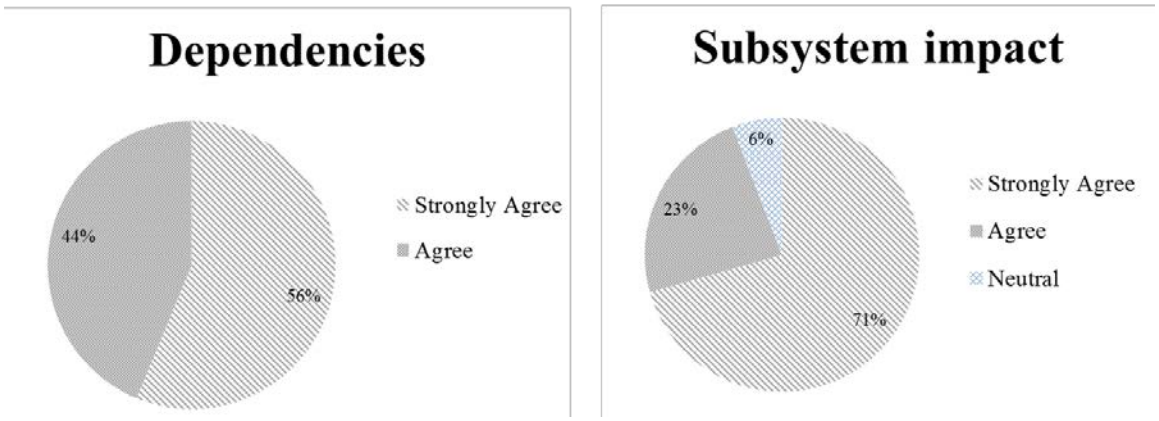


Figure 23: Survey results on the use of parameter dependencies' view and a subsystem impact view

When asked through an open question about other VM that they thought would be useful in a CE environment, one of the respondents answered that a 3-D physical view of the system could be especially useful for configuration purposes, another one mentioned the advantage of using a view to expose the evolution of the different design parameters throughout the design process.

In order to approach the performance of the design process, the survey attempted to grasp the timespan the designers thought would be acceptable to wait for the communication of the updated design parameters, and to optimize a design solution. The results were very different ranging from 5 to 3600 seconds for the first question, and from 10 to 600 minutes to the second one. This dispersion can be due to a biased perception at the subsystem level, for instance, the aerothermodynamics expert is used to make expensive calculations and,

therefore, considers it normal to wait longer than, for instance, the cost expert. The fact that the survey was performed with experienced users of CE environments can already somehow set a limit on the maximum duration of a specific calculation. In this context, designers could not foresee that any calculation (even those related to complex optimization methods) would take longer than the time between two design sessions which, typically, is not more than one day.

The final question of the survey focused on the optimization needs, the respondents were asked to evaluate, in their opinion, the adequate levels of optimization and fidelity needed in a CE environment. The designers were asked to place their choice in a matrix with 9 possible positions resulting from the combinations of 3 levels of fidelity and 3 levels of optimization (Figure 24 shows the distribution of the results in number of respondents).

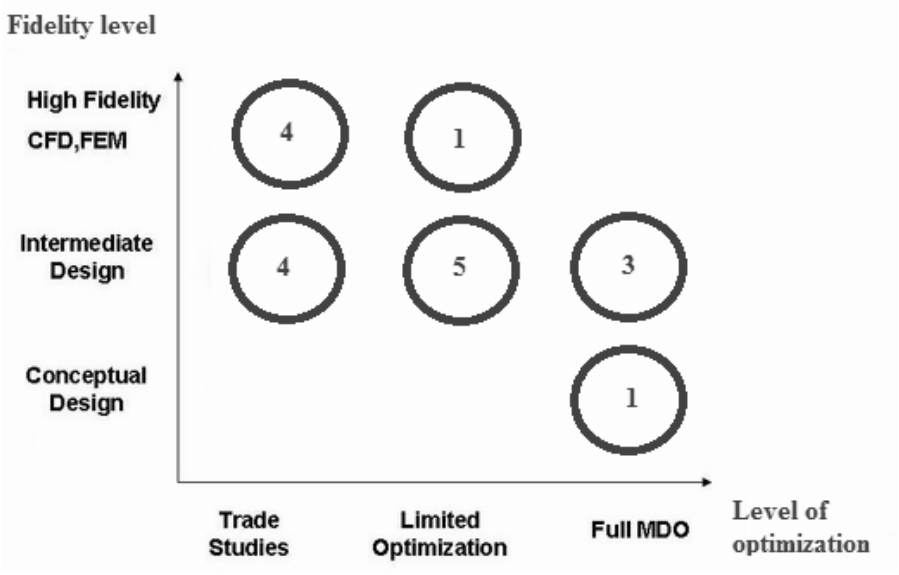


Figure 24: Survey results on the desired levels of optimization and fidelity in a CE environment

These results showed that most of the respondents perceived that the level of optimization in a CE environment should be restrained to trade studies or high level optimization, and should have an intermediate level of fidelity (more detailed than high level conceptual design, but with less fidelity than very specific models).

3.2. Preliminary ideas to increase performance on conceptual design

The initial research approach used the results of the survey (detailed in §3.1) and the main issues identified in the literature review (detailed in Chapter 2) to define the initial ideas guiding the development of the contributions of this work. This process was, by itself, a

complex iterative task, involving conversations with experienced designers and IT specialists, the results are organized in the following sections according to the different fields identified in Chapter 2: design processes, VM, KM, design modeling methodologies, and CSCD tools.

3.2.1. Improving the CE design process

CE relies on an iterative development method based on the spiral process (§2.1) that allows the inclusion of creativity and increases the flexibility of the design process. Unfortunately, although bringing everyone together in the same place speeds up the process on the long-term by reducing the number of future rework, on the short-term it increases the breadth of the analysis at the early design phase, leading to the perception of low efficiency.

The correct perception of the design process status can be achieved by *keeping track of changes on design parameters and critical open issues*. Once all the designers are aware of the bottlenecks it is possible to join efforts and increase the design process performance by tackling the problems real-time as a team.

The action-centric perspective of the spiral design process severely increases the benefits of creativity on the design by not restraining how the process should evolve. However, it is a very inefficient process when dealing with highly coupled subsystems, because there are only a few rules on guiding the interaction amongst them. One idea would be to *establish clear rules defining the procedures for data exchange, justification of design decisions*, and possibly include some milestones or deliverables to mark critical decisions (similar to the V-Model approach) that would somehow help gaining further information on the evolution of the system and support design convergence.

3.2.2. Using VM at the conceptual design phase

Choosing the right visualization for the system is a challenging task which involves determining how its structure can be better represented and manipulated by, for example, hiding irrelevant parts, drawing causal paths, or establishing intuitive metaphors. The different system parts may be organized into a tree, a ring, a dimensional order, a set of clusters, or some other kind of configuration. It is up to the designer to choose which of these forms is better (Kemp & J. B. Tenenbaum 2008). The ultimate goal of using VM for designing complex engineering systems is to increase the insight into their behavior and support better design decisions in shorter periods of time. Achieving true understanding of

a complex system is not possible by a simple visualization of its parts or features. It requires acting upon it, changing it, and observing the impact on the design.

Through the use of VM, the designer should become aware of the system's behavior from different perspectives *by condensing different points of view* to achieve a deeper understanding and knowledge of the system. Since the simultaneous representation of all the characteristics of a system will generally result in a lack of clarity, the VM should focus on specific aspects such as filtering and/or organizing information about the system in order to enhance features, problems, or other issues. The creation of this abstraction layer should be accomplished intuitively without expensive computations but, it is also important to provide the possibility of exploring lower level issues, focusing on the grassroots of each analysis through a «white box» approach.

This multi-layer approach should enable focusing on the «forest» instead of just the «tree», i.e. allow the visualization of the global system instead of just the various subsystems, possibly making apparent the hidden interconnections between the different domains of the system (Gil et al. 2010).

The designer should also be able to *interact with the system as a whole and become aware of the impact of his subsystem*. For example, if a value of a design parameter in a subsystem is changed, a representation of which other subsystems will be affected by this change can be crucial to keep the design in good track.

3.2.2.1. Visualization methods classification

Before selecting amongst the different VM (such as the ones introduced in section 2.2), it is important to *establish dimensions on which they can be classified*, one of the ideas is to organize them according to (Figure 25):

- Architecture representations suitability (e.g. a flow architecture is suitable to be represented by a flow chart).
- Objective, since some VM are only suitable for visualization (like static tables) while others can be more suitable for interaction (i.e. manipulation).
- Type of items that are more suitable to be portrayed, either data (quantitative data in schematic form) or information (where data is treated to amplify cognition).

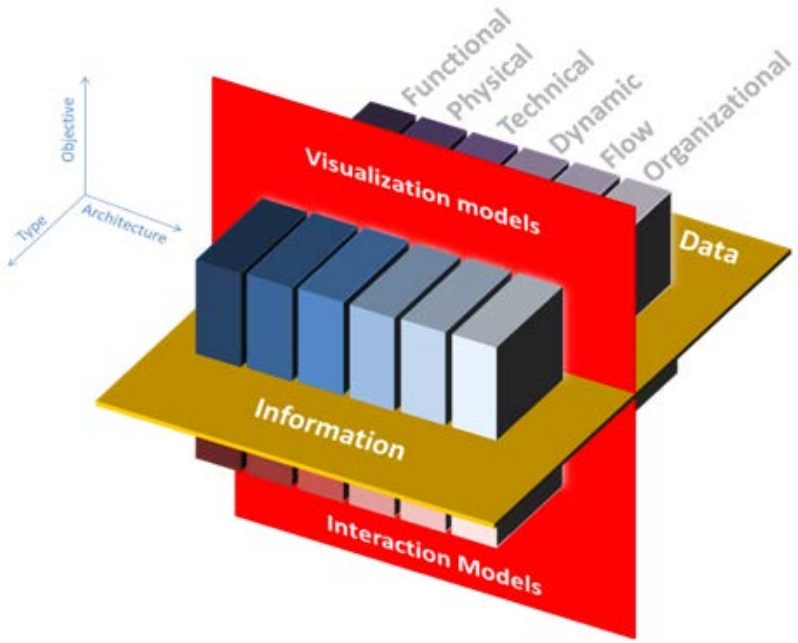


Figure 25: Classification of VM into three dimensions: type, architecture and objective

3.2.2.2. Visualization methods selection

VM can significantly increase the performance of complex systems design (T. Simpson et al. 2007) but it is necessary to proceed with care because unfortunate choices in the representation can make the process and user interaction too complex to be useful (Rhyne 2009). The problem of representing, visualizing, and manipulating a complex system is multidisciplinary, with an art component: to be successful it is *necessary to have input from designers about their needs*. These need to be gathered through unbiased tests and/or surveys designed to measure the insight gained by the use of a specific visualization on a specific task. Although surveys can be used for simple non-interactive visualizations, the measure of the suitability of visual metaphors should be developed through practical experiments because they are always highly dependent on how hard it is to grasp the human-machine interaction before actually trying it.

In order to keep these experiments objective, it is important to *not provide any hints on the choice of a specific viewpoint over another*. One of the possible approaches is to not instruct designers in exactly what insights they should be looking for, but let researchers observe what insights designers gain on their own (North 2006).

The *selection of the visualization methods should also depend on the design stage*, since some methods are more suitable for some stages than others (Figure 26).

		Visualization Models									
		Data Models				Information Models					
	Activities	Tables (a)	Cartesian coordinates graphs (b)	Scatter plots	Box plots	Cycle Diagrams	Timelines (f)	Data flow diagrams (c)	Radar charts	Information Lenses (a)	Tree maps (b)
Define Objectives	Define broad objectives and constraints		Op.					Op.			Op.
	Estimate quantitative mission needs and requirements	Op.	Op.							Op.	Op.
	Project Planning					Op.	Def.	Op.			
	Knowledge Base										
Characterize the Mission	Define alternative mission concepts	Op.	Def.	Op.					Op.	Op.	Def.
	Define alternative mission architectures	Op.	Op.	Op.				Def.	Op.	Op.	Op.
	Identify system drivers for each		Def.	Def.							Def.
	Characterize mission concepts and architectures	Op.	Op.	Op.	Op.			Op.		Op.	Op.
	Project Planning					Op.	Def.	Op.			
	Knowledge Base										

Figure 26: Example of VM selection according to the different design phases of space systems as defined by (Larson 1997) (Op. = Optional, Def. =Default)

3.2.3. Using KM at the conceptual design phase

Designing a complex engineering system is a highly demanding task from the perspective of the people involved in the process. This is probably the reason why it is so difficult to build knowledge bases for this type of systems; designers are usually too busy to allow themselves to be distracted from their central role in the design to document things properly. Furthermore, they usually do not see, or are not sensible to, the need of documentation because it does not directly affect their immediate work. One of the possible ways to solve this problem is to *ease the KM tasks as much as possible.*

It is important to allow *storing certain information automatically in an intuitive way* by using templates, automatic filled fields, lists, etc. Using standardized knowledge input methods can help overcome the differences amongst the different types of knowledge that are considered relevant at the subsystem level, by establishing a common ground to also make sure that knowledge useful at the system level.

The creation of incentives is very important to achieve successful KM (Edwards 2001). It is important to consider the successful examples of distributed web-environment projects mentioned in §2.3, and establish a *reward system based on meritocracy and transparency* with peer recognition and direct rewards such as reduced workload, learning new skills, or even money. An initial incentive to enhance KM tasks in a CE environment can be as easy as removing the need to elaborate burdensome reports from scratch.

3.2.4. Using design modeling methods at the conceptual design phase

A good representation of a complex engineering system can only be achieved through adequate design modeling methodologies able to encapsulate all the relevant features and behaviors of the system into a comprehensive language. Rather than trying to define a new design modeling method which may not have the acceptability nor the advantage of being mature such as the already existing languages (§2.4), the main issue will be the selection of the most suitable one for the specific application in mind that is not too complex and is independent of context (universal language). One idea is to consider a *combination of several MBSE methods* to try to compensate the shortcomings of one method with the advantages of another.

The complexity is one of the ubiquitous disadvantages of the current design modeling methods (§2.4). This fact reduces their acceptability from the CE designer's community which is usually under strict time pressures and high technical workload. It is highly advantageous to *integrate the design modeling method into the tools already used by the designers*. The number of different metaphors in the different schemes and languages should be minimized and intuitively available in commonly used modeling tools such as worksheets.

It is also important to develop *guidelines determining which subsystems should be considered and specifying the required level of detail*. Special attention should be given to the architectural decomposition of the system, and definition of the different subsystems. The system shall be divided into subsystems, that can either be physical or not (e.g. cost or risk), according to the technicality of the required analysis and ensuring, as much as possible, a balanced workload (even amount of analysis per subsystem).

Besides the identification of relevant subsystems, their level of detail must also be specified. Improper simplification or too much detail of single aspects can lead to drawing incorrect conclusions that may result in an unfavorable impact on the system performance or make the system's perception unnecessarily difficult.

The acquisition and monitoring of dependency structures can help increase the performance of the design process, particularly in early phases at the conceptual design phase when non-quantified information has to be modeled. The use of methods such as the DSM can significantly increase the perception of the system's coupled behavior, but common applications do not use this method synchronously, requiring effort from the

designers to document their changes *a posteriori*. An effort should be done to *reduce the delay between the definition of dependency structures at the subsystem level and their representation in a «DSM-like view»*, preferably integrating it in commonly used tools such as worksheet editors.

The number of different components and possible dependencies in complex engineering systems require a *methodical information acquisition to encourage designers not to forget or neglect relevant information that can undermine the final system's performance*. As an illustration consider an antenna specialist for an observation satellite. He/she may consider the geometry of the antenna mechanical supports as static and disregard the thermal cycle to which they are subjected throughout a complete orbit. Technically it is not his/her responsibility to model the mechanical supports (they probably have to be handled by the structural specialists) but he/she will be indirectly impacted by the vibration and consequent misalignment of the antenna. This typical engineering approach to problems is supported by years of expertise performing rough approximations on which a system is divided into clear domains (physical or not) with no significant couplings amongst them. This is not the case for complex engineering systems that cannot be reduced to a sum of simpler parts that can easily be quantified and independently analyzed.

Quality and correctness can be assured right at the definition process by checks and iterations with an *acknowledging scheme*, where the subsystem making the changes has to have them approved by the other subsystems impacted by these changes. Once determined, agreements and decisions have to be clearly documented through intuitive KM tools.

3.2.5. Improving CSCD tools

A successful CSCD implementation in a CE environment should reduce the interoperability issues that are expected to arise from the use of a multitude of subsystem level design tools, by establishing an *integrated collaborative design environment with intuitive additional toolboxes* enabling design process monitoring, design selection and design optimization. The development of this integrated environment should focus on reducing the learning commitment by requiring shorter learning curves. This will eventually result in a higher penetration of the tool amongst designers.

CSCD tools should be able to contribute to the understanding of complex systems by *enabling heuristic operations* that find hidden relations between parts, point out critical system elements, and clusters. They should allow a real-time perspective on which improvements,

such as structural reorganization or reduction of irrelevant parameter feedbacks, can help increase the efficiency of the design process.

It is also important to allow *different possible VM* to adequately visualize the naturally arising structures or processes, as an overview or on a detailed perspective, emphasizing on supporting both the divergent and the convergent paths of the design process (focusing on the generation of design possibilities and selection, respectively). The number of possible VM should be enough to allow a proper representation of the system at all stages but should not be overwhelming (§2.2).

The *KM system should be intuitively included* in the integrated collaborative design environment with the ideas introduced in §3.2.3: to gather knowledge and assist the design, not repeat mistakes, remember lessons learned, and avoid reinventing already available solutions. If these feedback tasks are divided by the different designers, there is a risk of turning them into simple «housekeeping» activities of the knowledge base. Assigning a dedicated person to this task can help increase accountability and therefore productivity.

The designers should be *able to choose the level of optimization* within the integrated collaborative design environment, depending on the detail of the mathematical models that describe the behavior of the system. The level of optimization has to be evaluated on a case by case approach. In some cases, the MDO process should be automated and incorporate as many design variables as possible (full MDO), but in other cases, because this process is too complex (computationally heavy) and/or the system is still not properly defined and the goal is to evaluate different alternatives, a simple trade-off study (or tradespace exploration) might be more appropriate.

[*This page intentionally left blank*]

4. INTEGRATED DESIGN METHODOLOGY

This chapter introduces a methodology to approach the main shortcomings identified in literature (Chapter 2) and recognized through the survey conducted with experienced users of CE design environments (§3.1). The general idea is that the design properties of complex engineering systems can emerge after a proper identification of the system-level objectives, and consequent assessment of their impact at the subsystem level. The design process can be significantly improved if design changes are supported with judicious design decisions that are intuitively documented by the designer. The proposed methodology considers four distinct phases (Figure 27): an initial system description stage (phase 1) (§4.1), a requirements' mapping stage (phase 2) (§4.2), an interaction stage (phase 3) (§4.3), and an optimization stage (phase 4) (§4.4).

The methodology evolves from a simple objective and a set of requirements/constraints to a final design, through an evolutionary iterative process. It promotes a dynamic interactive evolution of objectives, requirements/constraints, subsystems, system states and design models.

The designers are not asked for a specific design point solution, but for a set of models and constraints that guide their design choices. The methodology ultimately finishes with the generation of a set of possible design solutions and, through a tradespace exploration strategy, the designers are asked to converge towards the final design. The goal is to make the parameter transactions and choices made throughout the conceptual phase as evident as possible, moving away from a «black-box» philosophy (where the decisions are masked in the tacit knowledge of the designer), and simultaneously increase the overall performance in terms of design time and/or rework costs.

In terms of project management, this process supports the concept of curatorship. It considers that the design of a complex engineering system does not emerge as a part of an anarchistic group effort, but as the result of a guided process. This guidance role is taken by an individual commonly referred to as system engineer (or a small team) who merges elements during the design process, realigns them to comply with the global goal of the project, and manages resources to fulfill design needs on a real-time basis.

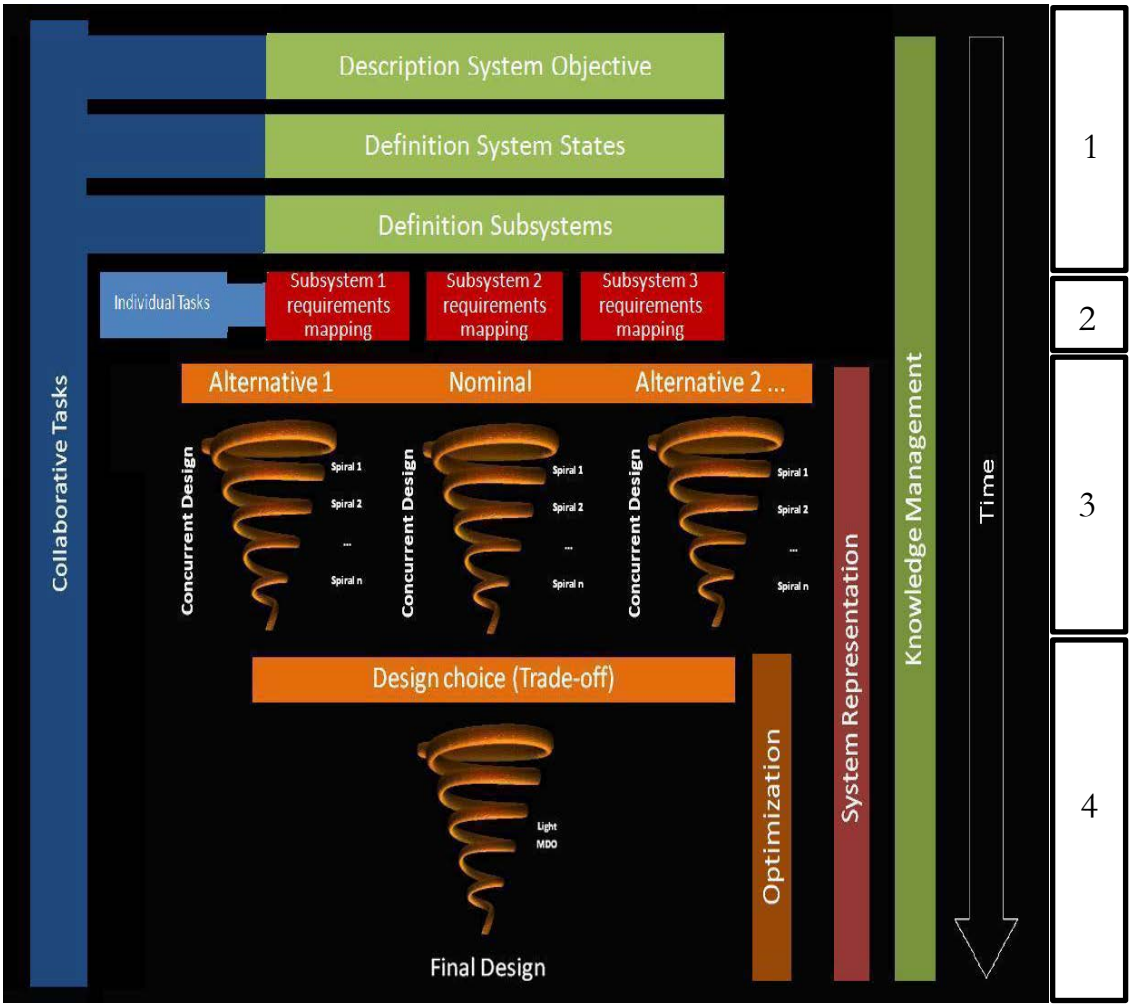


Figure 27: Integrated design methodology (1-initial system description stage, 2-requirements' mapping stage, 3-interaction stage, 4-optimization stage)

With the exception of the requirements' mapping stage, all the other stages are collaborative and require, somehow, CSCD tools to accurately integrate the different activities that need to be fulfilled.

The interaction stage might start with a set of alternatives, architecturally different, to include preference variability. These should be designed in parallel to arrive to a single option at the beginning of the optimization stage. The system description at the beginning of the optimization stage corresponds to a single architecture but, nevertheless, still allows attribute variability (§2.5.2.2). The final design (baseline solution) arrives after the optimization stage, where the different attribute values are decided.

The main ideas leading to this process were introduced in §3.2, the use of VM, the use of KM, and the design modeling methods. The way these guidelines are reflected into the different stages is described with more detail in sections 4.1 to 4.4.

Table 5: Map of the different guidelines (identified in §3.2) into the different stages of the integrated design methodology

Field	Guideline	Stages of the integrated design methodology
CE design process	Keep track of changes on design parameters and critical open issues	Interaction
	Establish clear rules defining the procedures for data exchange, justification of design decisions	Initial system description
Visualization methods	Condense different points of view	Interaction, optimization
	Interact with the system and become aware of the impact of its subsystem	Interaction
Knowledge Management	Ease the KM tasks as much as possible	Initial system description, requirements mapping, interaction, optimization
	Store certain information automatically in an intuitive way by using templates, automatic filled fields or lists	Initial system description, requirements mapping, interaction, optimization
	Reward system based on meritocracy and transparency	Interaction, optimization
Design modeling methods	Integrate the design modeling method into the tools already used by the designers	Interaction
	Guidelines determining which subsystems should be considered and specifying the required level of detail	Initial system description, requirements mapping
	Reduce the delay between the definition of dependency structures at the subsystem level and their representation in a DSM-like view	Interaction
	Methodical information acquisition to encourage designers not to forget or neglect relevant information that can undermine the final system's performance	Interaction, requirements mapping
	Acknowledging scheme	Interaction, optimization

A good representation of a complex engineering system can only be achieved by using design modeling methods that are able to encapsulate all the relevant behaviors of the system into a comprehensive language (Chapman 1992; Y. J. Chen et al. 2008). Rather than trying to define a new design modeling method which may not have the acceptability nor the advantage of being mature like the already existing languages (§3.2), the main issue will be the selection of the most suitable language that is not too complex (to shorten the necessary learning curve) and is independent of context (where the different metaphors remain consistent regardless of the representation chosen). The proposed methodology reuses two central concepts:

- that every system consists of by objects, processes, and states. Objects exist, and processes transform the objects by generating, consuming, or affecting them. States are used to describe objects, and are not standalone things. At any point in time, every object is in some state,
- the only commonalities at the conceptual design phase of any system are the existence of a (several) design objective(s) and requirements.

These key ideas are derived by combining and enhancing the ideas behind two existing languages: OPM (§2.4.2) and Harmony-SE (§2.4.1). Even though OPM considers a requirements' definition process, it does not intuitively allow transforming requirements into a system architecture consisting of objects, states and processes. Harmony-SE provides a service-request-driven perspective with a special focus of requirements traceability, but relies on SysML as the «systems algebra» which uses a high number of diagrams and is not easily understood by common system designers (that are not specialist in IT) (§2.4). This methodology proposes a combination of both to try to overcome their individual limitations.

The choices that led to the tools implemented in Chapters 5 and 6 are detailed in section 4.5, including their objectives and contributions within the integrated design methodology.

4.1. General system description stage

The integrated process starts by building a concept map around the system objective statement in order to clearly express what will be involved and is required to reach the objective (or mission statement (Larson 1997)). This will enable a graphic visualization of

the main goal of the system and, through a brainstorming activity, facilitate the convergence of ideas regarding the design objective.

Concept map technologies exist for organizing ideas at early stages and can add value to the engineering design process (Yuan Miao & Zhi-Qiang Liu 2000). The choice of specific conceptual mapping technologies to apply during this stage must take into account the level of expertise of the designers.

Figure 28 shows an example of a concept map to dissect a complex concept (telecommunications satellite) which can be better understood by establishing semantic relations between its inherited lower level concepts and grouping them into common attributes. Detailing the concepts expressed in the system objective in a set of simpler concepts can help understanding the scope of the design problem as well as identify critical design drivers.

A telecommunications satellite is, by itself, a combination of two complex concepts, telecommunication and satellite, which in their turn can also be decomposed into simpler concepts. Breaking down the concepts to simpler ones can lead to the emergence of some of the most important issues that will have to be tacked during the design process. In the case of a telecommunications satellite, the concept «link budget» relates to a lot of lower level concepts (orbit, data rate, bit error rate, attenuation and antenna), exposing its importance on the future design process.

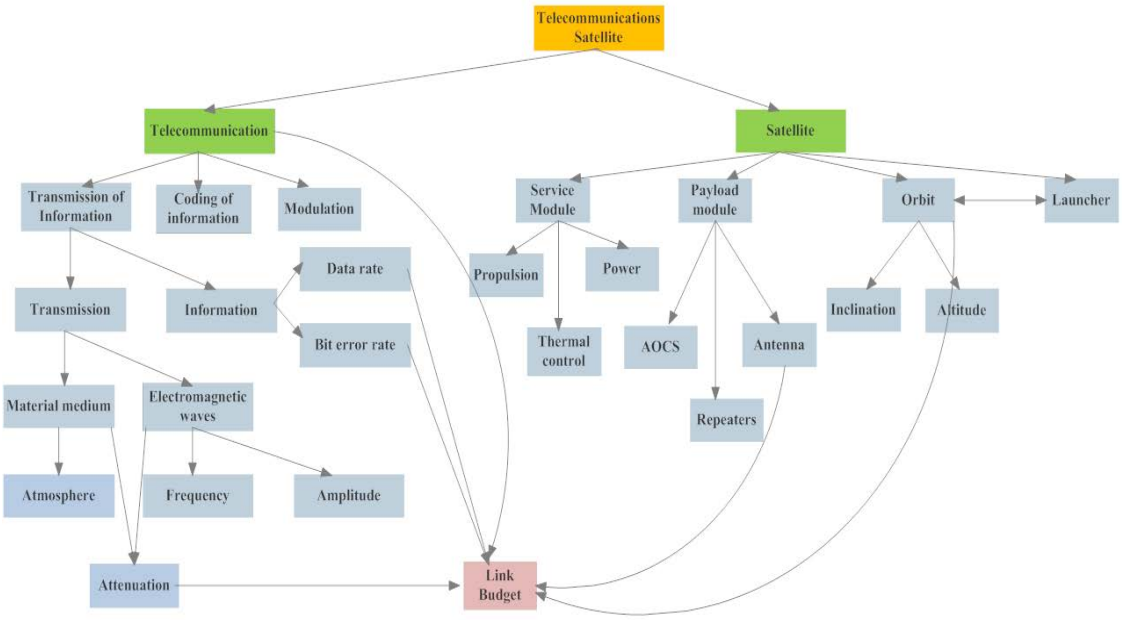


Figure 28: Example of a concept map of a telecommunications satellite. Relations between concepts such as the antenna, the orbit, and attenuation can contribute to the emergence of critical design parameters such as the link budget

The general description stage will continue with the definition of the different system states through which the system is going to pass during its lifecycle. This will help defining the drivers of the design. A state diagram can be very useful at this stage, to help represent the different design modes that will be the basis for the requirements mapping stage, without providing early design hints on a specific architecture, or design solution.

Figure 29 shows an example of the state definition diagram for a telecommunications satellite. It identifies all the states foreseen for the satellite throughout its lifecycle that will inherently influence the design process. The satellite will evolve from an «assembly mode» to an «end of life mode», passing through «launching and nominal modes», amongst others.

The development of this state decomposition in collaboration will foster the discussion into the expected functional behavior of the system. It can help to depict critical states (such as the «lift-off state» within the «launching mode» for the example of the telecommunications satellite), and sets a common ground for mapping future requirements without providing any hints on particular architectural solutions, e.g. mapping the structural load requirements to the «Max q» state will not provide any idea on the shape of the structure.

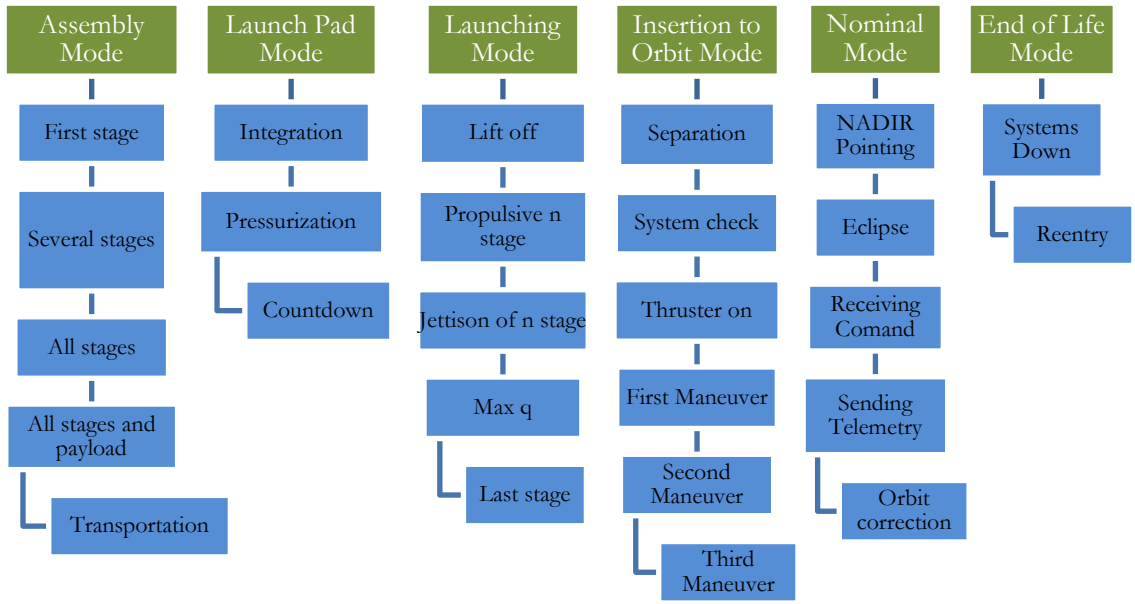


Figure 29: Example of a state definition for a telecommunications satellite. It describes all the states the satellite is expected to pass through its lifecycle and sets a common ground for mapping future requirements without providing any hints on any specific architecture

Depending on the state definition of the system and the emergence of system drivers, the initial description stage ends with the establishment of the different subsystems or areas of responsibility. These subsystems correspond to the physical and/or functional domains,

and are usually defined in a way that tries to minimize the interactions between subsystems (making a specific subsystem as hermetic as possible), by technical expertise, or other considerations. One example of this division is the domain definition in the context of space systems design (Bandecchi et al. 2000) where, in addition to physical domains such as structures or propulsion, also cost and risk can be turned into explicit domains.

The overall goal of the general description stage is to organize the initial design ideas by establishing a clearer process that evolves from a simple concept map to the definition of the different subsystems. This organized process will enable a convergence of ideas with respect to the design objective and main drivers that will guide future decisions. It can also be reused for future similar design processes as the initial iteration, avoiding extra work and facilitating the incorporation of all the previously acquired knowledge.

4.2. Requirements mapping stage

The subsystems will have their own specific requirements but will also necessarily have global overlapping requirements demanding interdependent analysis. The integrated design methodology should afterwards progress towards an individual subsystem stage where the different subsystem requirements (and also global high level system requirements) are mapped into the different system states previously defined.

Mapping the requirements to system states is highly beneficial from the design point of view because it does not provide any particular hints towards any particular design architecture, and avoids design convergence at an early stage, which could be a problem when they are mapped to a previously defined functional architecture. Every requirement has to be associated with at least one system state. An illustrative example can be found in Table 6, where Req.1 and Req.3 can be mapped into one single system state while Req.2 relates to two different system states.

An enhanced perspective is provided to the designer by mapping the requirements into the different system states. The goal is to enable the designer to clearly know which requirements (or other constraints) have to be considered when considering a specific system state. It will also provide a way to support future design decisions that will emerge during the other phases of the design process.

Table 6: Example of mapping requirements into systems states for a telecommunications satellite. For example Req. 1 must be verified during the nominal mode and, during the design process, should be taken into account by the subsystems: mission analysis, payload, AOCS, and propulsion

Requirements	Related system states	Subsystems associated
Req. 1 - The mission duration shall be longer than 10 years in normal operation	Nominal Mode	Mission Analysis, Payload, AOCS, Propulsion
Req. 2 - Bit Error Rate not exceeding 10^{-7}	Receiving and Sending Telemetry and Command Modes	Communications
Req. 3 - The separation manoeuvres shall be totally accomplished with satellite visibility from the ground stations	Separation mode	Mission Analysis, AOCS, Propulsion

The need to map requirements into different design modes has long been advocated as a best practice in the field of requirements engineering systems (Sommerville 1997) with fruitful examples in numerous applications ranging from building construction to automotive systems. In the aerospace field, Nichols (2008) proposed an organization of requirements into three major areas: launch vehicles, ground support equipment, and launch support equipment, which could be mapped into the different launch modes and configurations. Mapping requirements into design states helps traceability and increases design efficiency by linking requirements to design items, test plans, test cases or other inhered requirements.

4.3. Interaction stage

This is the stage during which both parameter dependencies and assignments are changed and justified. It corresponds to the typical CE process where the designers responsible for each subsystem work together to agree on the dependencies flow and perform design decisions. It is at this stage that: new dependencies can be established by requesting parameters from other subsystems, existing dependencies can have their status changed, the possible values for a specific parameter can be changed, etc. Every alteration can be supported by requirements or other design justifications.

Usually, DSM methodologies (§2.4.3) are used after the occurrence of the design phase to look at how the process evolved and possibly how it can be improved for the next projects (Avnet & Weigel 2010). It is an asynchronous analysis with no real-time feedback to the different subsystem specialists when the design is actually being defined.

This methodology proposes the use of a dynamic DSM to simultaneously provide a static view over the component-based architecture of the system (i.e. subsystems), and a time evolution view of the parameters that determine the design. The subsystems are represented in the diagonal entries, and the parameter dependencies in the off-diagonal entries. Every time there are any changes in any of the design attributes (items, parameters, assignments, etc.) they are immediately reflected into the structure of the DSM that is accessible to all the members of the design team (Figure 30).

Following the current shortcomings of this field identified in Chapter 2, and the survey performed to expert users detailed in Chapter 3, this methodology proposes monitoring five different design aspects throughout the interaction stage:

- keep track of *status of design parameter dependencies* at the system level, i.e. between the different subsystems (or domains) and their operational status (detailed in §4.3.1),
- display the different *assignment types* for the design parameters. These different assignment types typically act as design decisions and should be monitored throughout the design evolution (detailed in §4.3.2),
- inform on the *quality of the assignments* made. This can help emphasizing promising designs from initial guesses by providing a metric about the assumptions (detailed in §4.3.3),
- show the *impact of each parameter* on the downstream subsystem which requested it. Displaying this information during the design phase places the designer one step ahead of his/her decision, getting an idea where it will impact the most. It can help make critical design choices clearer and reduce burdensome calculations during reiterations, and therefore increases the performance of the concurrent phase. The impact clues can also play an important role in the parameter sequencing and design tradespace exploration by helping to establish in which order the assignments for the generation of the different design instances should be done (detailed in §4.3.4),

- display the different *parameter completion status*. This can help figuring out how far the design is from a final one and help depict potential problems or bottlenecks during the design process (detailed in §4.3.5).

This list of design aspects was chosen to encompass the most critical needs to gain insight into the status of the design process, without overwhelming the designer with too much information. This list can be extended or reduced for specific systems.

Changes in these design aspects can be monitored by using different metaphors such as color, font size or patterns. Several combinations of metaphors can be used to track different design aspects at the same time (Figure 30). The choice of the code to represent a specific status should be user-dependent to accommodate any cultural differences, e.g. a Japanese designer might prefer using green to represent a specific type of parameter assignment, while a European might choose, for the same type of assignment, the red color.

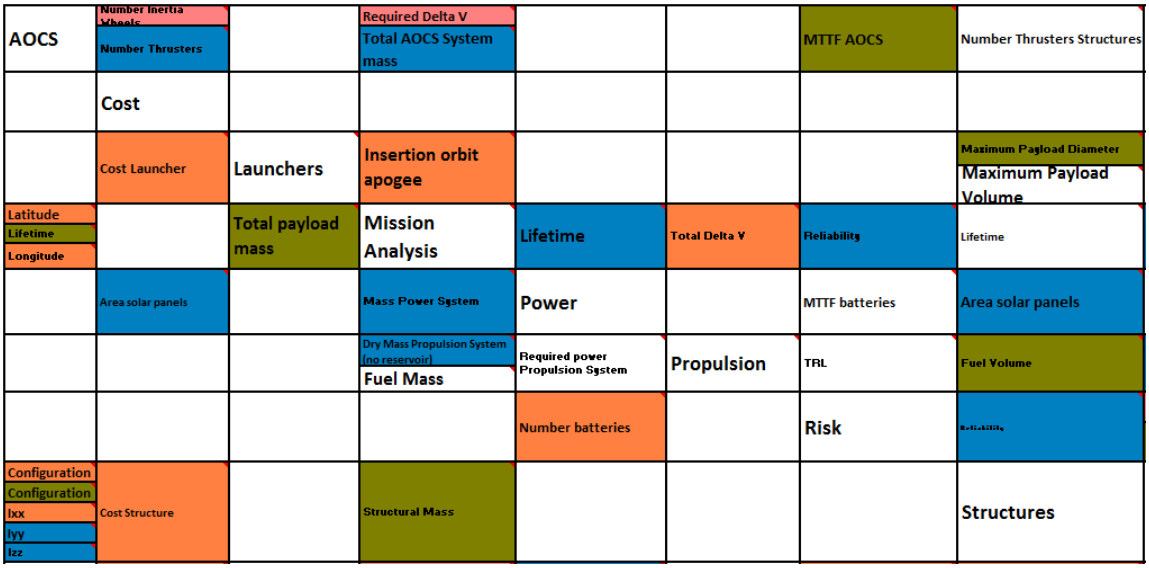


Figure 30: Example of a partial dynamic DSM view of a telecommunications satellite during the interaction stage. Subsystems are on the diagonal entries, and parameters on the off-diagonal entries. In this example, color is used to represent the different dependencies status and the font size the different parameter assignments. Other possibilities would be to represent the quality of the parameter assignments, the impact of these on the downstream subsystems, or the parameter completion status

The above aspects should be integrated in a KM strategy where the designer is able to document his/her design decisions with certain requirements, constraints, ad-hoc verifications or any other external files. The designer will be guided through the interaction stage to evolve from mapping the inter-subsystem relationships to observing the impact of a parameter on a specific subsystem.

Whenever any of the designers decides to change one of the states of these aspects, he/she is asked to include a justification that can be: a small ad-hoc text description, a «known» system or subsystem requirement, a design constraint, or an external file. This information will be saved, along with the change itself, to be able to easily understand the rationale behind the decision and keep track of design evolution, e.g. evolution of mass budget throughout a certain design session or several design iterations. This information can afterwards be leveraged for future similar design applications.

4.3.1. Parameter dependency status

This aspect provides information on the status of the parameter requests and is defined by both the designer that asks for the parameter (input domain), and the designer that delivers the parameter (output domain) (Table 7). On a normal operation, the former starts by requesting a new parameter from an output subsystem/domain, and the latter will decide if there is an acceptance of this request or not. Once there is an acceptance, the status will depend on the output switch provided by the output, it can be internal if the parameter is not yet shared with the input subsystem/domain, manual if it is shared but there is no explicit model defining its value, or external if it is shared and there is an underlying explicit model defined at the output subsystem/domain level.

Table 7: Examples of possible status that can be tracked for parameter dependencies during the conceptual design phase

Possible states	Description	Defined by
Requested	The parameter has been requested by the input subsystem/domain	Input subsystem/domain
Rejected	The parameter has been rejected by the output subsystem/domain (reiteration needed by the input subsystem/domain)	Output subsystem/domain
Internal	The parameter has been accepted but is not yet shared by the Output subsystem/domain	Output subsystem/domain
Manual	The parameter has been accepted and shared by the Output subsystem/domain, but there is no underlying model defining its value	Output subsystem/domain
External	The parameter has been accepted and shared by the Output subsystem/domain and there is an underlying model defining its value	Output subsystem/domain

4.3.2. Assignment types

This aspect provides information on the type of assignment chosen by the output subsystem/domain. It will reflect the considerations that the designer has on the overall possible values given to this parameter. The designer will dispose of several possibilities depending on whether there are certain constraints to a fixed point assignment, multiple discrete options, possible range, or no constraints at all (Table 8).

Table 8: Examples of types of parameter assignments that can be tracked during the conceptual design phase

Possible states	Description	Defined by
Not defined	The parameter is not manual nor external yet, or, the output subsystem/domain did not decide on the assignment type yet	Output subsystem/domain
Free	The output subsystem/domain cannot elaborate any considerations on the possible parameter values. This state is only available if the parameter dependencies status is external.	Output subsystem/domain
Range	The output subsystem/domain established an upper and a lower bound to the possible parameter values. This state is only available if the parameter dependencies status is external.	Output subsystem/domain
Multiple	The output subsystem/domain established several alternatives to the possible parameter values. This state is only available if the parameter dependencies status is external.	Output subsystem/domain
Point	The output subsystem/domain fixed a parameter value. This state is only available if the parameter dependencies status is manual or external.	Output subsystem/domain

It is expected that different design possibilities will arise when the designer chooses the multiple or range assignment types. Nevertheless, some design alternatives are radically different or imply choices of more than one parameter and therefore cannot be represented just by choosing the multiple or range assignment types, e.g. choosing a communication network either by a constellation of satellites in LEO (Low Earth Orbit) or, one single GEO (Geosynchronous Orbit) satellite. In this case, the incorporation of these alternatives can sometimes be eased by defining separate alternative systems rather than using the same architecture with multiple or range assignment types.

4.3.3. Assignment quality

This aspect provides information on the maturity (level of confidence) of the assignments made. It provides a qualitative assessment of the confidence on the values provided and indirectly ascertains the possibility or a parameter assignment reiteration. Rather than providing a scale of values, the designer is asked to place the quality of the assignment in a set of predefined ranks (Table 9).

This is a direct metric of the depth of the analysis done. If the assignment quality is low this value is nothing more than an order of magnitude or an initial guess and is, therefore, likely to change throughout the design process. If the assignment quality is high, the assignment was defined with the help of a well-established model or rule and is, therefore, less likely to change throughout the design process.

Table 9: Examples of the qualities of assignments that can be tracked during the conceptual design phase

Possible states	Description	Defined by
Not defined	The assignment is not defined or the output subsystem/domain did not decide on the quality of the assignment yet.	Output subsystem/domain
Low	The assignment was made with large assumptions and is very likely to change.	Output subsystem/domain
Medium	The assignment was made with some assumptions and is likely to change if the other subsystems request it.	Output subsystem/domain
High	The assignment was made with very few assumptions and not likely to change. A change should be made by a system request rather than a subsystem one.	Output subsystem/domain

4.3.4. Parameter impact

This aspect provides a qualitative assessment of the influence of each parameter on a subsystem/domain. It will ultimately allow the designer to be one step ahead of his decision by measuring the changes inflicted by changing a certain parameter, on another subsystem other than its own domain on which he/she is not specialist. Rather than providing a scale of values, the designer is asked to place the impact of the parameter in a set of predefined ranks (Table 10).

The overall subsystem definition should be evaluated with a high level regard on all the subsystem parameters (both the domain specific, not shared, and the ones shared with other subsystems). The most affected subsystem parameter is the one that, amongst the ones shared with the other subsystems, changes the most with a change in the input parameter in question.

Table 10: Examples of the levels of impact of assignments that can be tracked during the conceptual design phase

Possible states	Description	Defined by
Not defined	The parameter is not manual nor external yet, or, the output subsystem/domain did not decide on the impact of the parameter yet	Input subsystem/domain
None	The parameter has no impact on the subsystem/domain.	Input subsystem/domain
Low	The parameter has little impact on the subsystem/domain. The «overall subsystem definition» does not change more than 10%, or the most affected subsystem parameter does not change more than 30% for all the possibilities of this parameter.	Input subsystem/domain
Medium	The parameter has medium impact on the subsystem/domain. The «overall subsystem definition» changes between 10% and 30%, or the most affected subsystem parameter changes between 30% and 50% for all the possibilities of this parameter.	Input subsystem/domain
High	The parameter has high impact on the subsystem/domain. The «overall subsystem definition» changes more than 30%, or the most affected subsystem parameter changes more than 50% for all the possibilities of this parameter.	Input subsystem/domain

4.3.5. Parameter completion status

Another critical aspect that is important to trace during the concurrent iterations is the design completion status. This will help knowing how far the current design is from the final one and will help depicting potential problems or bottlenecks in the process itself.

This work proposes that, when the designer includes a justification for a change in a parameter state, he/she also chooses between one of the states presented in Table 11.

Table 11: Examples of the completion status that can be tracked for the different parameters during the conceptual design phase

Possible states	Description	Defined by
Not completed requiring iteration	One of the subsystems/domains made a change and the next step requires feedback from the other subsystems/domains that relate with that parameter	Input or Output subsystem/domain
Not completed not requiring iteration	One of the subsystems/domains made a change and the next step only depends on itself	Input or Output subsystem/domain
Completed	The parameter definition reached its final status (no other changes in the parameter state are expected)	Input or Output subsystem/domain
Problem	There is a problem on the agreement of the parameter state (amongst the subsystems/domains involved). Requires intervention at the system level.	Input or Output subsystem/domain

4.4. Optimization stage

During the conceptual design phase the existence of an optimization stage is highly dependent on the quality of the subsystem level models, and the desired depth of analysis. It can range from simple trade-offs of different system configurations for design selection purposes (§2.5.2.2), to highly elaborate MDO methods (§2.5.2.3). Instead of a static (final) point solution, the outcome of this stage is a baseline design solution that establishes the possible range of architectures for posterior, more detailed, design studies.

Diller (2002) compiled a list of common problems referenced in literature when optimizing during the early design stage:

- establishing design requirements a priori with limited consideration of other options,
- inadequate means of systematically evaluating broad trades in the early stages of design,
- lack of regard for the complete preferences of the decision maker,
- inaccurate characterization of the decision maker preferences, pursuit of a detailed design without understanding the effects on the larger system,

- limited incorporation of interdisciplinary expert opinion and diverse stakeholder interests.

Optimization at the conceptual phase can increase the performance of the design phase by better framing the design baseline solution and therefore avoid useless work to detail suboptimal designs during future design phases. However, it should be done in a judicious way to avoid the pitfalls detailed earlier. This methodology provides a series of guidelines to increase the transparency of the decision path and «move away» from standard «black box» optimization practices.

This methodology considers that the optimization stage is used to narrow down the baseline design solution. During the last part of the interaction stage, the designer is able to understand which parameters have more impact on individual subsystems and can now establish a weighted DSM sequencing operation to minimize feedback couplings. Feedbacks imply additional iterations. Irrelevant couplings should be eliminated where possible (S. D. Eppinger et al. 1994).

Properly sequencing the design process facilitates the generation of a large set of feasible designs, by reducing parameter feedbacks. One of the possibilities for the sequencing algorithm is to use a heuristic method to define a parameter impact metric (§4.3.4), and rank first the subsystems that have more influence on the others (and are less influenced by the others).

Figure 31 describes a possible heuristic algorithm, where W represents the weighted DSM with normalized parameter impacts from 0 to 1, I^n represents the impact of n-level parameter dependencies (e.g. a second level dependency occurs when the output of one subsystem influences a subsystem which, in its turn, influences another one), and Z is the sum of all the impact contributions of the different levels of parameter dependencies.

$$Z = \lim_{n \to \infty} \sum I^n$$

It is possible to quantify the global influence of one subsystem on the others by adding the values of Z that are placed on the same row of the subsystem, e.g. the global impact of subsystem A can be estimated by adding the values in the first row of Z.

$$\text{Relative impact of subsystem } i \text{ on the others } (M) = \sum_{j=0}^{n} Z_{ij}$$

In order to quantify how much a specific subsystem is influenced by the others, the algorithm just adds the values of Z on the same column of the subsystem, e.g. the effect of the other subsystems on subsystem B can be estimated by adding the values in the same column.

$$\text{Relative Impact on how subsystem } j \text{ is impacted by the others } (N) = \sum_{i=0}^{n} Z_{ij}$$

Obviously, the subsystems which influence more the others and are less influenced by them should be evaluated first, in order to minimize the number of iterations. Therefore the subsystem ranking is done according to the higher ratios between M and N.

$$\text{Ranking criteria} = \frac{M}{N}$$

$$W = \begin{matrix} \begin{bmatrix} 0 & 0.1 & 0.7 \\ 0 & 0 & 0.3 \\ 0.3 & 1 & 0 \end{bmatrix} \begin{matrix} A \\ B \\ C \end{matrix} \\ \begin{matrix} A & B & C \end{matrix} \end{matrix} \qquad \begin{cases} I^1 = W \\ I^n = (I^{n-1}W).\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{cases}$$

$$I^1 = \begin{bmatrix} 0 & 0.1 & 0.7 \\ 0 & 0 & 0.3 \\ 0.3 & 1 & 0 \end{bmatrix} I^2 = \begin{bmatrix} 0 & 0.70 & 0.03 \\ 0.09 & 0 & 0 \\ 0 & 0.03 & 0 \end{bmatrix} I^3 = \begin{bmatrix} 0 & 0.03 & 0.21 \\ 0 & 0 & 0.063 \\ 0 & 0 & 0 \end{bmatrix}$$

$$I^4 = \begin{bmatrix} 0 & 0.21 & 0.09 \\ 0.0019 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad I^5 = \begin{bmatrix} 0 & 0.009 & 0.063 \\ 0 & 0 & 0.0132 \\ 0 & 0 & 0 \end{bmatrix} \quad I^n = \cdots$$

$$Z = \lim_{n\to\infty} \sum I^n \cong \begin{matrix} \begin{bmatrix} 0 & 1.143 & 1.043 \\ 0.114 & 0 & 0.380 \\ 0.300 & 1.030 & 0 \end{bmatrix} \begin{matrix} 2.186 \\ 0.494 \\ 1.330 \end{matrix} \\ \begin{matrix} 0.414 & 2.173 & 1.423 \end{matrix} \end{matrix}$$

$$R_A = \frac{2.186}{0.414} = 5.280 \quad R_B = \frac{0.494}{2.173} = 0.227 \quad R_C = \frac{1.330}{1.423} = 0.935$$

$$Sequence: A \rightarrow C \rightarrow B$$

Figure 31: Example of a heuristic method to sequence subsystems. I^n is the n-level weighted dependency matrix generated from the first level impact matrix (W). It estimates the relative impact of each subsystem on the others to rank first the subsystems which are less influenced by the others and that influence more the others

More sophisticated algorithms can be envisaged to test the different sequencing possibilities through genetic algorithms (Yassine 2006), or other stochastic methods.

Design feasibility is defined at the subsystem level, where the different assignments made by each subsystem limit the number of possible designs. A design is only feasible if all its subsystems are feasible, and comply with the overall system requirements.

Once the design process is sequenced, it is necessary to choose the type of optimization. There are a number of issues that need to be considered when choosing the type of optimization:

- level of detail, depending on the quality of the subsystem-level models and the desired quality of the baseline solution,
- number of possible designs, increasing the number of designs generated helps guarantee that an optimum solution is chosen, provided that the representation of these solutions does not overwhelm the designer.
- calculation time, obviously, the lower the calculation time the better, since the designer can use this time to perform more productive work such as detailing subsystem level models, or reconsider other revolutionary designs. Since the CE sessions are usually constrained in terms of time, there needs to be a compromise between the number of design possibilities generated and the calculation time.
- flexibility, the type of optimization chosen should be flexible enough so that the problem formulation, applied constraints, and the level of simulation can all be specified by the design team.

This methodology considers three types of optimization methods: simple trade-offs, tradespace exploration (§2.5.2.2), or more detailed MDO (§2.5.2.3). Table 12 assesses the characteristics of these methods with respect to the previous issues: level of detail, number of possible designs generated, calculation time and flexibility.

Table 12: Evaluation of different types of optimization to be used during the conceptual design phase

Type of optimization	Level of detail	Number of possible designs generated	Calculation time	Flexibility
Trade-off	Low	Low	Low	High
Tradespace	Medium	High	Medium	Medium
MDO	High	Medium	High	Low

This methodology requires a highly dynamic environment, where the definition of the system and formulation of the optimization problem (constraints and objective functions) constantly change throughout the design process. The algorithms chosen for optimization purposes need to be highly flexible, intuitive, and decomposable into different calculation processes (allowing multiprocessor calculations and/or utilization of recent cloud-based calculation resources). The possibilities are endless, ranging from random search to

complex constrained optimization methods (Figure 32). The characteristics and choice of the different optimization algorithms is beyond the scope of this work.

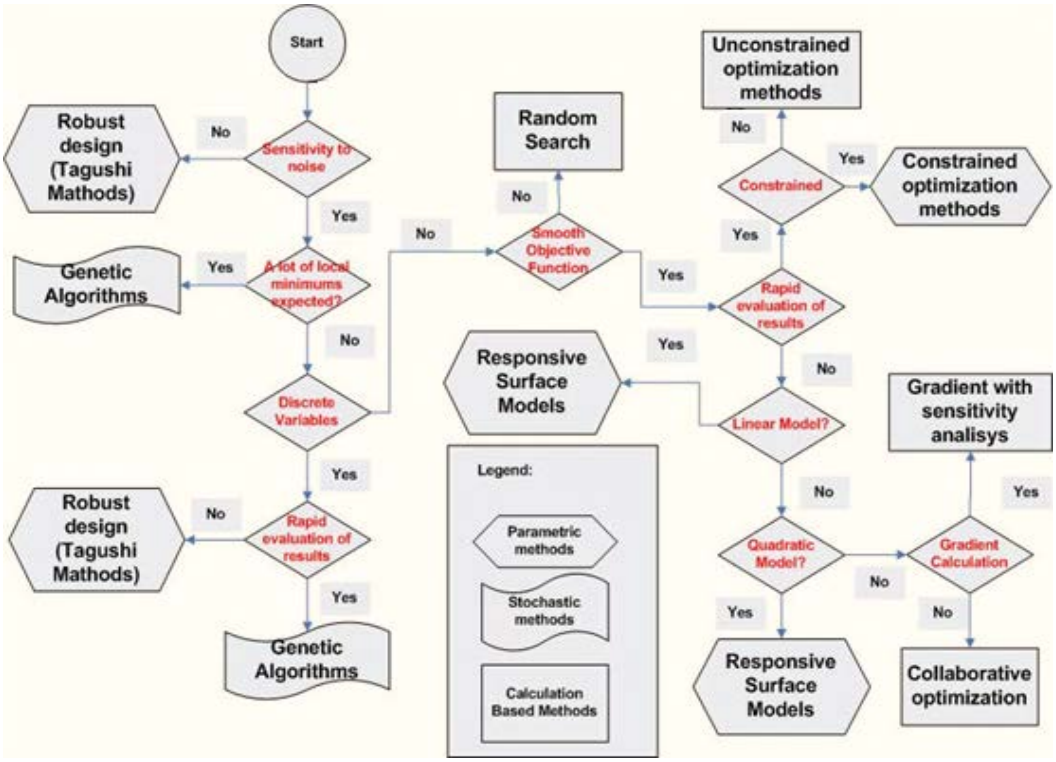


Figure 32: Example of procedure to choose the optimization algorithm for a specific problem, adapted from (Pedregal 2003)

Once the different optimum design possibilities are calculated, the designer can navigate through them to prevent incoherent results, or intuition-biased optimization paths. Instead of being hidden inside a «black box», the optimization path remains transparent by keeping the designer inside the decisions' loop.

Finally, in order to evaluate the quality of the solution, the designers can be led to define a decision process based on the AHP method (§2.5.1.1), a multiattribute utility function dependent on a selected number of parameters (§2.5.1.2), or an agent based prediction support tool (§2.5.1.3) to help converge towards a baseline design solution.

4.5. From the methodology to the implementation

A methodology such as the one described in the earlier sections can only be supported through a set of CSCD tools that allow its implementation. Developing tools to test and measure the impact of all the methods proposed throughout the different stages of the proposed integrated design methodology would be an insurmountable task within the

scope of this work. It is important to clearly define the fields of possible implementations to obtain measurable results.

Regarding the general description stage, there are already numerous tools able to implement concept maps, some of the most prominent are (SmartDraw n.d.) or (Cmap n.d.). The state-like view of the system can already be approached by MBSE tools that support UML or SysML, such as (MagicDraw n.d.) or (Papyrus n.d.).

There are also tools that can be applied during the requirements mapping stage, one of the most commonly used in industry and academic fields is (IBM DOORS n.d.). These tools can provide an easy way to map requirements into the different system-level for future traceability of decisions by engaging all stakeholders in a collaborative requirements process.

Therefore, this work focused on two specific tools to improve the needs of, mainly, the interaction and optimization stages: an agent-based decision support tool, and a higher level tool named INtegrated Concurrent Real-time EnvironMENT (INCREMENT). The following chapters detail these two software tools which were developed to support the most prevailing capabilities introduced in the proposed methodology.

5. PREDICTIVE DECISION SUPPORT TOOL

The conceptual design phase should be, by definition, relatively short (§1.2). Approaching it in a CE environment increases its flexibility and performance, but also establishes stricter time constraints. Gathering a group of highly skilled experts in the same time frame (often even in the same space) involves a considerable amount of logistics. It is important to concentrate these scarce resources to achieve a global successful baseline solution, usually, in a matter of a few design sessions.

Much of the work performed during these CE design sessions involves developing non-deterministic subsystem models, based on previous similar design solutions. Developing accurate historic models during the conceptual design phase can yield a significant gain in terms of the quality of the baseline solution, but requires precious modeling time from the different designers. These models are a critical part of the design process but are often specific to one subsystem and therefore do not need to be done concurrently.

Developing tools that help reduce the time required to accurately model these phenomena can increase the performance of decision support by saving important design time, which can then be used for system-level tasks that require interaction between the different subsystems.

Computers are commonly used to model a variety of engineering activities, but the main focus of computer applications are currently in areas with well-defined rules, where the goal is known from the beginning (Paliwal & Kumar 2009). Activities related to the conceptual stage of the design process are generally not ruled by computers because they are often seen as *ad-hoc* processes where human experience, i.e. tacit knowledge, still plays a key role (Ulrich, 2004).

IT-enabled tools can help dealing with the common limitations of «human-based» decision processes such as recurring to associative thoughts (comparing the current solution with previous experiences), or complexity abstraction (inability to cope with high level coupled behaviors). This work analyzed 3 types of computer aided collaborative decision support methods (§2.5.1): the Analytic Hierarchy Process, the Multi Attribute Utility Analysis, and the agent-based modeling methods.

The first two methods are especially tailored for choice and ranking activities, dealing with multicriteria and/or multiobjective problems by enlightening the decision process through

measurable weights or utility functions. They are powerful methods requiring low calculation resources focusing on very high level (almost strategic) decisions. They do not need to be implemented in dedicated tools, although several exist both for web-based (AHPproject n.d.) or standalone applications (Ross et al. 2004). The application of such methods during the conceptual design phase has been the subject of research works conducted by Saaty (2009) and Malak Jr, Aughenbaugh, and Paredis (2009) who have focused on developing customized tools and measuring the impact of these methods to achieve better overall design solutions.

Historically, due to an insufficient support of dynamic collaborative design environments, agent-based modeling has been detached from CE environments where the decisions usually involve complex and non-deterministic interactions that evolve during the design (§2.5.1.3) (Shen 2001). These methods have proved to be very effective in modeling complex behaviors during detailed design but are not specifically tailored for the initial design phase.

Section 5.1 introduces how agent-based modeling can be used to reduce the turnaround time of typical conceptual design iterations by cutting on the time required for modeling non-deterministic behaviors, without compromising the dynamic evolution of the design process.

Enhanced decision support tools can improve the quality of the design process by assisting the designers engaged in decision-making activities to find a solution that better suits their goal, within shorter periods of time. Their role is particularly important on both selection and compromise decisions, during the interaction and optimization stages of the integrated design methodology defined in the sections 4.3 and 4.4. The selection decisions focus on reducing the number of design alternatives into a realistic and manageable number based on different measures of merit. The compromise decisions emphasize on making appropriate tradeoffs based on criteria relevant to the feasibility and performance of the system.

5.1. Using agent-based modeling for decision support

Agent-based modeling uses autonomous simple agents to predict the appearance of complex phenomena. It is a multi-scale approach where a simple set of rules describing the

action of individual agents (micro-level) allows the emergence of complex macro level behaviors (§2.5.1.3).

The agent characteristics, such as autonomy, ability to perceive the surrounding environment, act in specialized domains, and their capability to cooperate with other agents, made these methods useful in applications like decision support. By using a loosely coupled network of problem solvers, multi-agent models are able to solve problems that are beyond their individual capabilities (Shen et al. 2008). Multi-agent models work by learning, adaptation and reproduction rules specifically targeted at prediction activities, i.e. using previous experience to forecast possible results not yet experienced.

Design models are the core of every design process. They establish an abstraction of reality to better understand the different phenomena of a particular system. There are two basic approaches to construct these models: a deductive and an inductive one (Saaty 2009). The deductive approach models the global system into subsystems ruled by known rules (mathematical and/or physical laws). The inductive approach supports the use of estimation models from measured data, in order to find patterns that can be exploited to predict behaviors and, therefore, explain the different phenomena.

During the detailed design phase the deductive approach is the one usually chosen. However, during the early stages of a design process, the inductive approach also plays an important role because the design architecture is still not mature enough, and/or there is not enough knowledge on the system to express all its phenomena with deterministic rules. The designer often recurs to the inductive approach using approximation techniques. These techniques use previous design solutions to generate results which are accurate enough to produce a reliable design solution.

The inductive approach requires a great deal of analysis which tend to increase exponentially with the complexity of the system, and drives the conceptual design phase away from a concurrent environment on which the different experts are supposed to quickly generate new designs and grasp the feasibility of a certain design solution. Being able to generate an approximate value or simply an order of magnitude for every design variable is of extreme importance during the conceptual design phase. The development of such models is a time-consuming task (Abyaneh, Karegar, & Al-Dabbagh, 2004; Huthwaite, 1988), whose quality can also be undermined due to the high number of assumptions made or an erroneous choice of the approximation model (Cooper, 2004).

Using agent-based models can help to improve the performance of inductive approaches by providing accurate design suggestions in a short period of time. By defining only a few set of rules for individual agents, it is possible to mimic complex behaviors and therefore provide useful predictions for a specific design solution.

The increasing number of agent-based model applications led to a wide range of models ranging from static to adaptive Bayesian networks. The definition of the characteristics of all these different models is out of the scope of this work and can be found, for instance, in work of Shen (2001). Rather than an exhaustive review, this work focuses on some of the most prominent agent-based methods: Artifical Neural Network (ANN) models.

5.2. Artificial Neural Networks

An Artificial Neural Network is a mathematical model network that simulates the behavior of biological neural networks in order to learn from previous actions to predict complex behaviors.

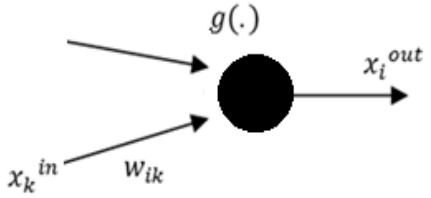
The prediction capabilities of ANN can help reduce the time required for *ad-hoc* inductive models during the interaction and optimization phases of the conceptual design. Currently, there are numerous software solutions to implement ANN, ranging from simple Matlab packages, to open-source standalone products, but their use is still overlooked in CE environments due to an arrangement of factors: their long learning curve (the methods require specific knowledge to make network architectural choices), selective applicability (the methods are usually tailored for specific applications), and lack of information on the quality of the solution provided by the algorithm.

ANN models are currently applied in numerous areas ranging from health prediction-monitoring systems (Akay et al. 2011) to financial analysis (Enke & Thawornwong 2005). The early historical roots of the ANN field can be traced to the work of Lapicque (1907). Their inherited abilities such as nonlinear learning and noise tolerance make them particularly useful in situations where the problem is likely to change. ANN can be especially useful when solving problems that cannot be clearly represented with a procedure, i.e. expressed as a series of steps, such as recognizing patterns, classification, series prediction and data mining (Arbib 1995; Heaton 2008; Vellido et al. 1999).

An ANN is obtained by combining by simple processing elements, called neurons (Figure 33) that can exhibit complex global behavior (Arbib 1995; Hopfield & Tank 1985),

determined by the connections between the processing elements and element parameters. The «integrate and fire» model of the neuron was introduced as early as 1907 (Lapicque 1907) and, due to its simplicity, is still one of the most popular models in agent-based modeling.

Making informed decisions during the early stages of the design process requires a great deal of analysis, usually involving *ad-hoc* approximation methods to provide a reliable evaluation of expected behaviors. Unfortunately, the time spent on developing these models decreases the available effective development time, and therefore reduces the performance of the conceptual design phase. Using a predictive decision support tool based on ANN models can enhance the decision-making process during the conceptual design phase by using information from previous design solutions. Hence, the focus is on a supervised learning paradigm on which each data sample (previous design solution) consisting of independent system variables X (factors) and dependent performance variables Y (responses) are used to update the ANN definition.



$$\boldsymbol{x_i^{out}} = g\left(\sum_{k=1}^{n} w_{ik}\, x_k^{in} - v_i\right)$$

$\boldsymbol{x_k^{in}}$ = input k of neuron i

$\boldsymbol{w_{ik}}$ = weight input k of neuron i

$\boldsymbol{g}$ = activation function

$\boldsymbol{x_i^{out}}$ = output neuron

$\boldsymbol{v_i}$ = threshold neuron i

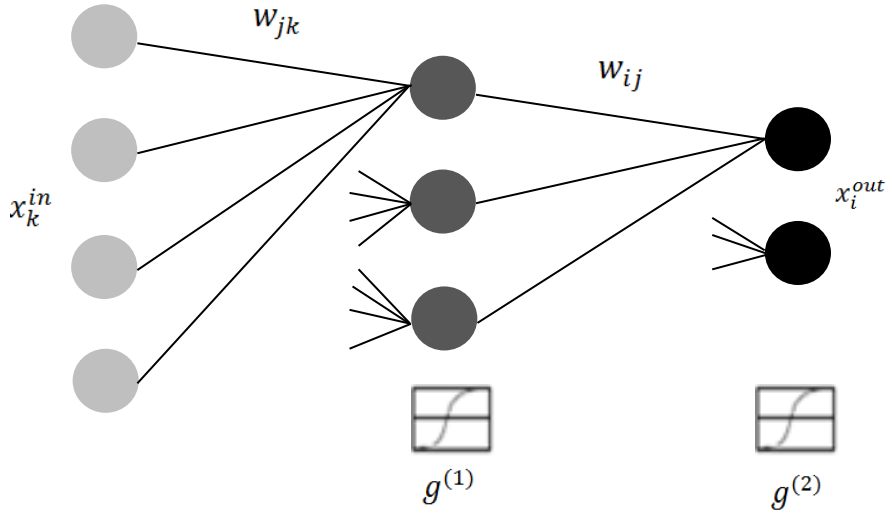
Figure 33: Integrate and fire model of the neuron

The ANN definition process has two phases: the training and the testing (Arbib 1995; Steeb 2008). During the training phase, the input dataset is exposed to the ANN and weights are adjusted to fit the training data as much as possible (learning it). In the testing phase, the data points which were not used during the training phase are compared to the outputs of the ANN to evaluate its performance. In spite of the advantages listed above, ANN models often need significantly large training datasets to achieve accurate design suggestions and require the definition of numerous parameters that are not straightforward from the perspective of non-specialists. Nevertheless, they outperform other automatic approximation methods based on Design of Experiments (DOE) techniques, such as the Response Surface Methodology (RSM) (Box 1987), both in terms of lower complexity, and higher accuracy, especially when dealing with non-linear problems (Shen 2001). An architecture that consists of a simple neuron receiving input from a number of channels is

called a simple perceptron. This type of architecture can only solve linearly separable problems and, therefore, exhibits reduced performance in applications that require robust adaptation to the different dataset types. A multilayer perceptron can deal with nonlinear problems by enabling a smart pre-processing phase resulting in a linearly separable problem feeding into an output perceptron (Arbib 1995; Heaton 2008).

5.2.1. Multilayer feedforward perceptron architecture

Multilayer feedforward perceptron models have enjoyed a privileged position in the neural networks community because of their simplicity, approximating power, and various historical reasons (Heaton 2008). In a multilayer feedforward perceptron architecture, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) to the output nodes (Figure 34). There are no cycles or loops in the network. This architecture was the one selected to develop the predictive decision support tool based due to its high quality and speed performances when solving nonlinear problems (Lim et al. 2000; Paliwal & Kumar 2009; Vellido et al. 1999).



$$x_i^{out} = g^{(2)}\left[h_i^{(2)}\right] = g^{(2)}\left[\sum_j w_{ij} x_j^{hid}\right]$$

$$= g^{(2)}\left[\sum_j w_{ij} g^{(1)} h_j^{(1)}\right]$$

$$= g^{(2)}\left[\sum_j w_{ij} g^{(1)}\left(\sum_k w_{jk} x_k^{in}\right)\right]$$

Figure 34: Multilayer feedforward perceptron architecture

Most applications use Multilayer feedforward perceptron models with a single hidden layer, adding more hidden layers allows the network to extract higher-order statistics and parametric behaviors but requires higher computational power. The network acquires a global perspective despite its local connectivity due to the extra set of synaptic connections and the extra dimension of neuron interconnections.

5.3. Neural network methodology

This work intends to promote the use of ANN models to support design decision during the conceptual design phase by:

- Defining an intuitive tool that can be used from the perspective of designers without any background in ANN or agent-based modeling,
- Developing a robust algorithm that can be automatically applied to a large range of applications, abstracting its complexity from the perspective of the designer, and,
- Providing a quantitative metric on the accuracy of the solution given.

A feedforward multilayer architecture was selected to assist during the conceptual design phase by selecting suggestions of both discrete and continuous parameters. The number of layers and neurons per layer were left as design parameters that need to be defined by the user. A supervised learning technique was selected to use previous design solutions as training data, and a Levenberg-Marquardt algorithm (Kisi 2004; Marquardt 1963) to update the weight values until a satisfactory error is reached.

The weight update has been carried out by solely regarding the training error, defined as the mean squared error (MSE) for the training dataset, i.e.

$$E_{training}(w) = \frac{1}{2}\sum_{\theta}(t^{out}(\theta) - x^{out}(\theta, w))^2$$

Where $E_{training}$ is the training error, w a specific combination of weights, t^{out} the expected output, and x^{out} the neural network output, and θ a sample in the training dataset.

The most common algorithms for weight update are the gradient descent and the Newton methods. Nevertheless, these algorithms behave poorly under certain conditions. The first method can cause problems due to the linear search approach and has the disadvantage of being relatively slow close to the minimum. The Newton method has a quadratic

convergence rate but does not perform well when the initial guess is far from the optimum and it may cause problems if the tangent is parallel or nearly parallel to the search vector (Arbib 1995).

The Levenberg-Marquardt method has been used to overcome the above problems. In this method a small constant value is added to the diagonal of the Hessian matrix. The larger the value, the more it behaves as the gradient descent method. After a few iterations the value added to the Hessian matrix is reduced (to zero) and it achieves the convergence rate of the Newton's Method.

5.3.1. Software architecture

The decision support tool consists of three main processes: the first treats the training data and normalizes it, the second defines the ANN, and the third uses the ANN applied to specific inputs to assist on a particular design decision (Figure 35).

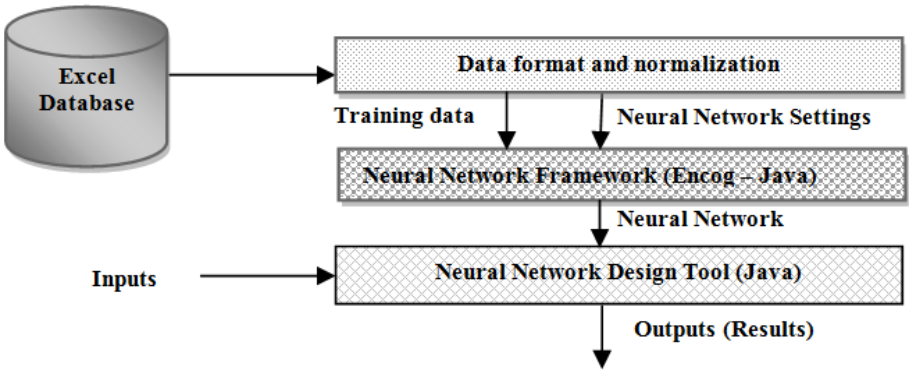


Figure 35: Structure of the decision support tool

The program starts by collecting raw data from a database in the form of a simple spreadsheet. After being formatted and normalized it will serve as both training and validation data. Three format types were considered: continuous, discrete and Boolean variables (Figure 36).

The normalization procedure consists of transforming the inputs (from the three types mentioned above) into values belonging to the [0,1] interval. This procedure corresponds to:

- a linear interpolation (considering the maximum and minimum values, and scaling parameters) for continuous inputs,
- direct correspondence of true = 1 and false = 0 for Boolean inputs, and,
- an integer vector for discrete variables. In this case, neurons are created in an equal number to the number of possible discrete events, i.e. if a discrete input can be A,

B or C, three neurons are created, if the input is A the output of the normalization procedure will be 1 for the corresponding neuron and 0 to the neurons B and C.

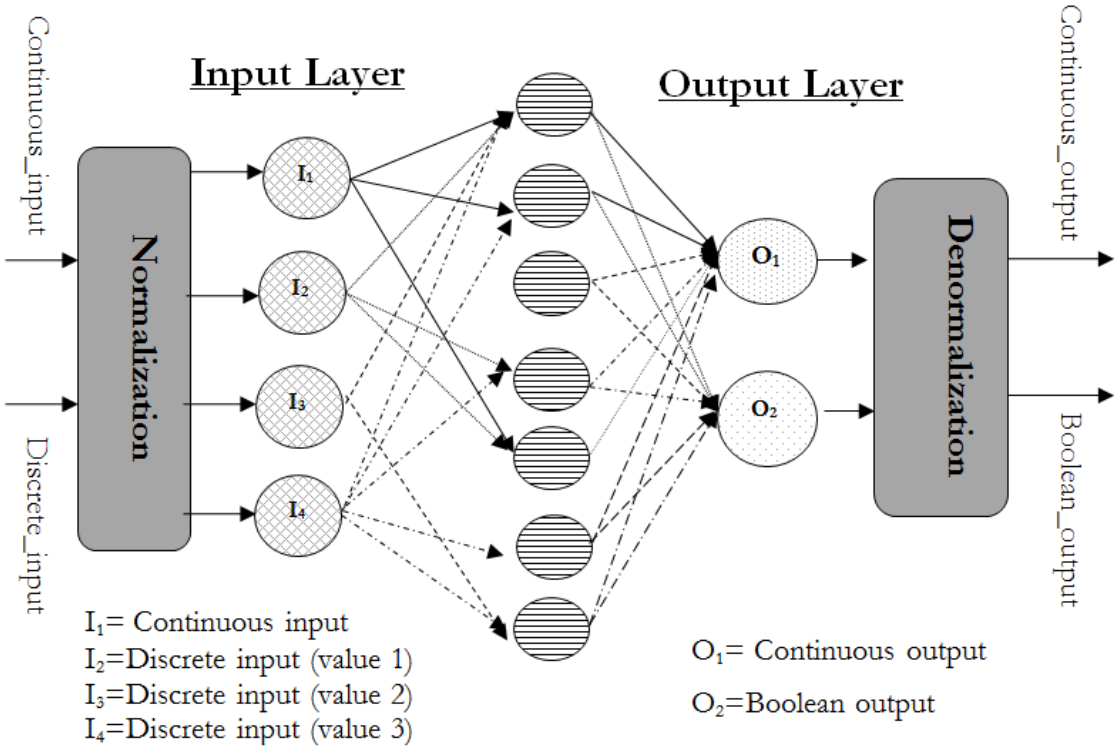


Figure 36: Structure of neural network for an example with two types of inputs (one continuous and one discrete) and two types of outputs (one continuous and one boolean)

The neural network was developed with the help of an open source neural network Java–based package Encog (Heaton 2008). The database information and all the other inputs required by the neural network were set with the help of a software tool described in section 5.4.

5.3.2. Design choices

The predictive decision support tool was designed to be used in a large array of potential applications, from continuous single variable to discrete multivariate problems. Currently, there are many tools capable of implementing neural network analysis in different types of problems, but they usually require a long learning curve to understand both the theory of neural networks, and/or the use of the software (Steeb 2008). This acts as a barrier for the non-expert user requiring a prompt meaningful suggestion for a specific design problem. Additionally, there are other critical barriers even for those that venture in this learning process: typically the existing tools establish data interface requirements demanding a considerable effort from the user; the selection of the neural architecture is often overwhelming and difficult to master; finally, once the calculation is performed, there is no

easy way to visualize and export the output. The developed tool is an attempt to detach, as much as possible, the user from the neural network design. This is accomplished by providing an accessible interface with the data worksheets, intuitive data mining and filtering procedures, freezing the best compromise for neural architecture, providing intuitive stopping conditions, and including multi-parameter output visualization capabilities.

Activation functions (§5.2) are the source of nonlinearity in the neuron model. Nonlinear activation functions can turn multilayer perceptron architectures into nonlinear function approximations (Arbib 1995). In the decision support tool a sigmoid activation function was selected due to its adaptability for both discrete and continuous problems (Asaduzzaman et al. 2009). The advantages of applying this activation function were confirmed during early performance tests.

Establishing the stopping conditions for the algorithm is one of the critical issues when developing a neural network analysis tool (Hopfield & Tank 1985). The developed tool offers several possibilities to define the stopping condition for the algorithm: set a time limit, set a limit in the number of calculations, stop when the training error achieves a certain value, stop when the testing error achieves a certain value, or stop when training vs. testing error divergence is found. The first four conditions can be used simultaneously, e.g. the user can set a time limit and a training error value and the calculation stops whenever one of these conditions occurs, or when both occur. The last condition tries to optimize the training time by detecting the point where there starts to be a divergence between the training and the testing errors, showing that any further weight changes will result in function over-fitting, a phenomena known as the «bias/variance dilemma» (Geman et al. 1992).

The «bias/variance dilemma» is one of the most serious problems that arise in connectionism learning by neural networks. It consists on the over-fitting of the training samples and happens when model fits very closely the training data, but does not generalize well, i.e. it cannot model sufficiently well unseen data from the same set. Since the ultimate goal behind machine learning is attaining generalization, over-fitting problems should be detected and addressed carefully.

The implemented training vs. testing error divergence detection rule aimed to dynamically approach the «bias/variance dilemma» by averaging every a certain number of epochs with training and testing errors (in this case 10), and comparing them in sets of consecutive

averaged samples (in this case 10) (Figure 37). Whenever the difference between two consecutive averaged values (e.g. $A_{31}^{40} - A_{21}^{30}$) passes a certain threshold (for this application, -1×10^{-9} for the training error, and 1×10^{6} for the testing error), a «divergent event» is triggered (black boxes in Figure 37), showing that the training error convergence was achieved, and the testing error already started to diverge, i.e. the ANN performs well when matching the data it was trained for but it started to exhibit worse results for data it was not trained for.

Once the number of «divergent events» reaches at least half of the size of the «comparison window» (in this case, when it reaches at least 5), the stopping event is activated, finishing the training procedure. The choice of both the size of the comparison windows, and the limits (training and testing) was made through a set of early trials to define a compromise between the divergence detection and the extra required calculation time (5 minutes was selected as the limit for the convergence of each run), this value can be changed in the tool but was hard-coded to reduce complexity from the user's perspective.

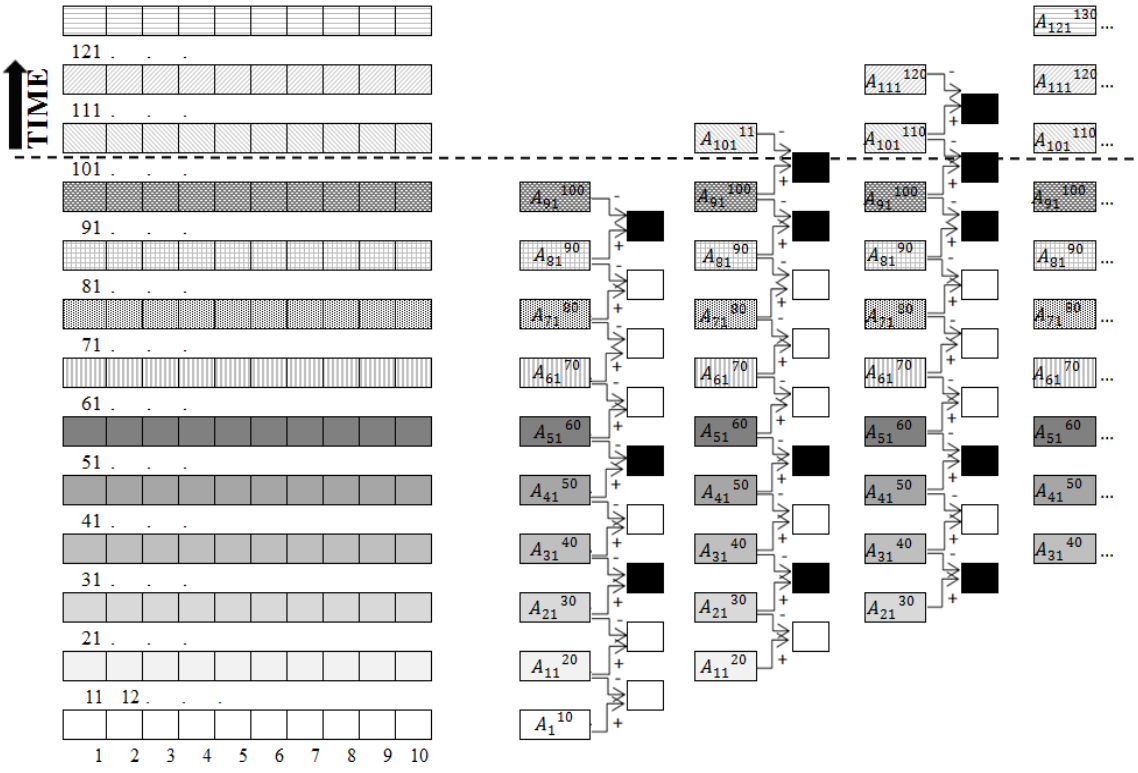


Figure 37: Training vs. testing error divergence detection rule. A_1^{10} is the average value of the samples 1 to 10. The algorithm looks at a «window» of 10 averaged samples, whenever there are more than 5 samples where the training error does not converge and the testing error diverges, the algorithm stops in order to avoid over-fitting («bias/variance dilemma»)

5.4. Implementation

The decision support tool was implemented in Java as a standalone application. A graphical interface was developed to reduce the learning curve to a minimum. The user is guided through a set of tabs that that are interchanged automatically with the problem definition.

The first tab is the initial data definition tab (Figure 38), where the excel file path is defined and data definition can either be done automatically (by specifying only the initial column, row and tab), or manually (by selecting certain columns and/or rows).

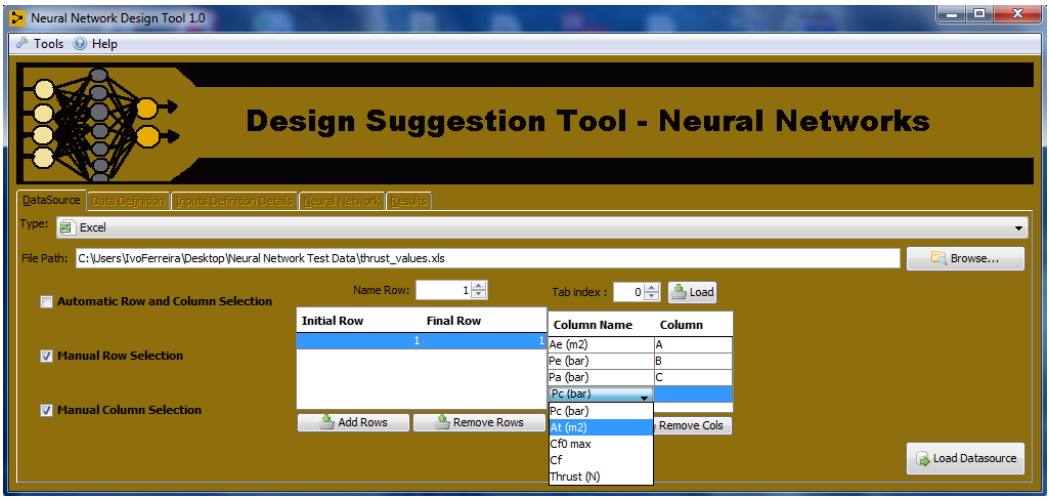


Figure 38: Initial data definition tab in the decision support tool

The second tab is the data filtering tab (Figure 39), allowing the definition of subsets of the initial data in the excel file (for the training procedure) by enabling continuous and discrete filtering conditions.

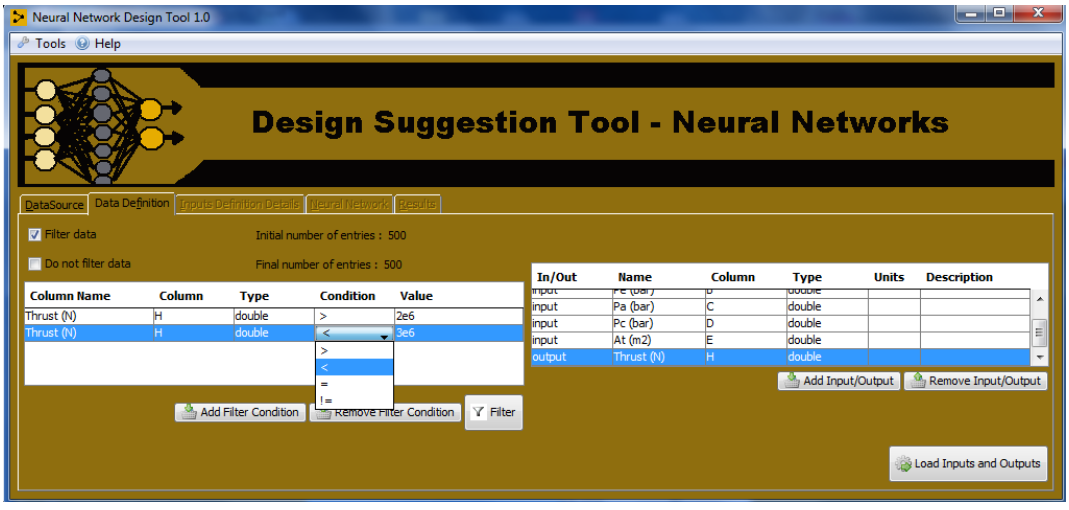


Figure 39: Data filtering tab for in the decision support tool

The next tap is the neural network architecture and training tab (Figure 40), allowing: the choice of the stopping condition for the algorithm (either based in time constraints,

number of iterations, training error, testing error, or training/testing divergence detection rule - Figure 37); the definition of the percentage of the dataset used for testing error calculation, the selection of the number of hidden layers, and neurons per layer.

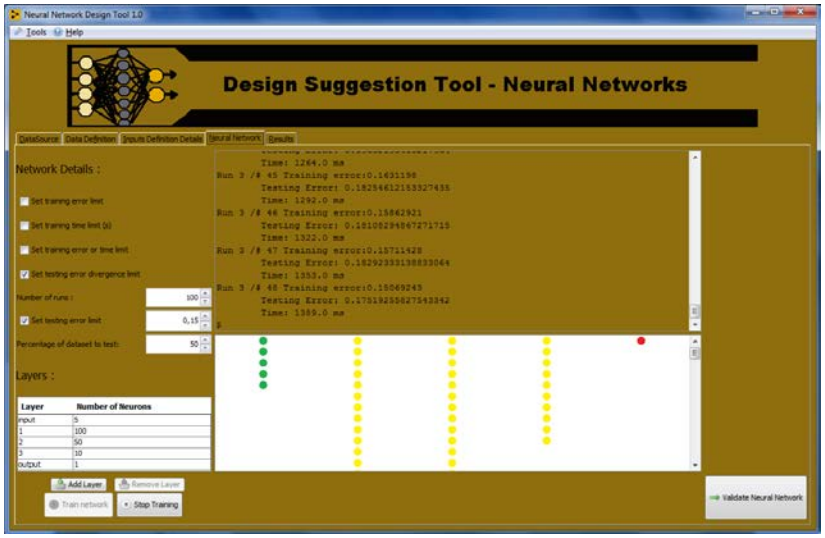


Figure 40: Neural network architecture and training tab in the decision support tool

Finally there is a the results tab (Figure 41), allowing point design analysis or a global evaluation for different ranges of input parameters. A dynamic parallel coordinates graph is used to allow a quick assessment of the relations between parameters and possible ranges. It also provides a data exploratory environment with the ability to rescale and reorder the different axis.

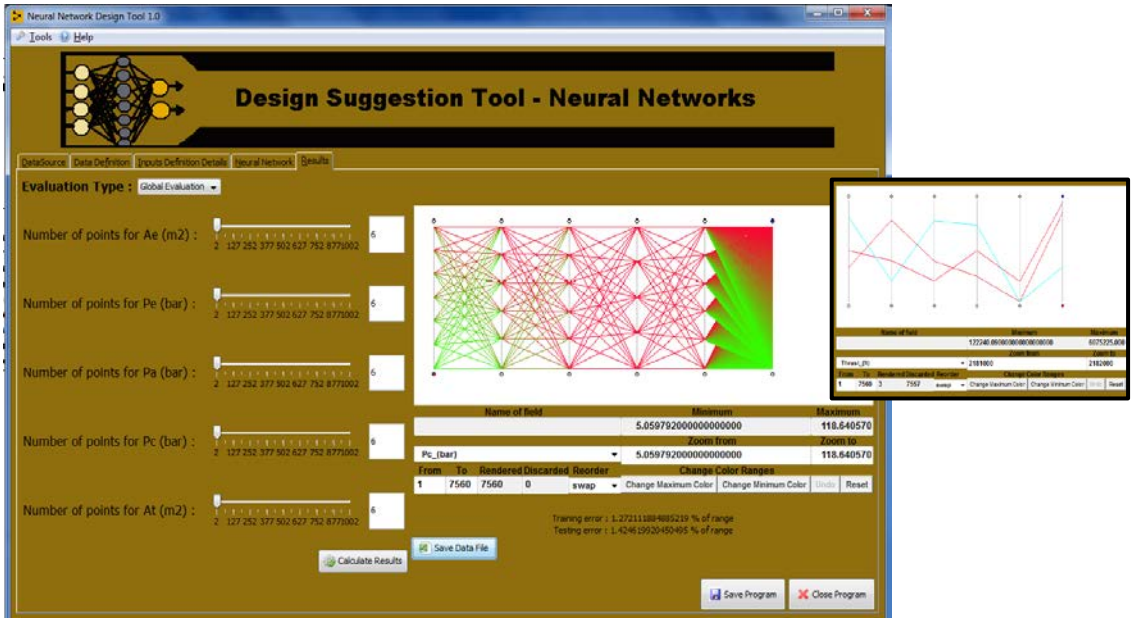


Figure 41: Results tab in the decision support tool

The quality of the solution can be assessed by displaying the training and testing error. Finally, the user is offered the capability of exporting the results to a simple spreadsheet file for further analysis.

5.5. Performance tests

One of the key requirements for a decision support tool tailored to the conceptual design phase is high adaptability. It is important to obtain meaningful results in an intuitive and swift way for a large array of problems. This was one of the main drivers for this decision support tool that resulted in a series of architectural choices to achieve a robust instrument that can be applied, as effortlessly as possible, to problems with different levels of complexity.

In order to test the performance of the tool, nine different examples of models were used to assess the prediction error for different types of functions ranging from simple linear to highly non-linear functions (Table 13).

Table 13: Functions used for the performance tests of the predictive decision support tool. These functions were selected for their variability and diverse behaviors (represented in Appendix 2)

Function number	Type	Function
1	Linear	$3x$
2	Quadratic	$5x^2$
3	Cubic	$-7(x-0.5)^3$
4	Log	$3\log(10x)$
5	Logistic	$\frac{50}{1+72^{-\frac{x}{5}}}$
6	Exponential	$e^{\frac{x}{50}}$
7	Periodic	$40\sin\left(\frac{8\pi x}{100}\right)$
8	Non-linear	$\frac{0.01x^3}{e^{\frac{x}{5}}-1}$
9	Non-linear and periodic	$\frac{0.01x^3}{e^{\frac{x}{5}}-1}40\sin\left(\frac{8\pi x}{100}\right)$

These performance tests helped determining the influence of the different issues related with the neural architecture definition, or the dataset distribution, on the quality of the predicted solution. The idea was to follow a Design of Experiments (DOE) orthogonal approach with the assumption that the number of neurons, the number of samples in the

database, the number of trials, the data distribution and error are independent parameters, uncorrelated in terms of their influence on the performance of the tool. With this simplification in mind, the effects of these issues were tested one by one (§5.5.1 to §5.5.4) while keeping the other parameters unchanged. This was performed in order to identify their individual contributions into the overall performance of the decision support tool.

The performance was measured by comparing the predicted results obtained with the tool against the expected results that were not part of the ANN training dataset. Two quantitative measures were used: the Root Mean Square (RMS) of the error and the coefficient of determination (R^2):

$$RMS = \frac{\sqrt{\sum_\theta (t^{out}(\theta) - x^{out}(\theta,w))^2}}{n} = \frac{2\,E_{training}(w)}{n}$$

$$R^2 = 1 - \frac{\sum_\theta (t^{out}(\theta) - x^{out}(\theta))^2}{\sum_\theta (t^{out}(\theta) - \overline{(t^{out}(\theta))})^2} = 1 - \left(\frac{RMS}{\sigma[t^{out}(\theta)]}\right)^2$$

Where t^{out} is the expected output, and x^{out} the neural network output, $E_{training}$ the training error, w a specific combination of weights, θ a sample in the training dataset, n the total number of samples in the training dataset, and $\sigma[t^{out}(\theta)]$ the standard deviation of the expected output.

During the tests, the predictive decision support tool performed generally very well with most of functions with R^2 values between 0.8 and 0.999. The only exceptions were with the periodic functions, where the accuracy was compromised by the poor performance of ANN models in approaching this type of functions. In the following sections the overall results of the performance tests are presented. The graphs with the results for the different individual functions can be found in Appendix 2.

5.5.1. Influence of the ANN architecture

Using ANN models to support design decisions requires special attention regarding several issues related to its architecture. The more pervasive structural choices are isolated from the designer and fixed to ensure accuracy and robustness: the use of a multilayer perceptron architecture, sigmoid activation functions, and the Levenberg-Marquardt for the update of the neuron weights (§5.3.2). Nevertheless, some of the parameters can still be customized depending on the type of problem, namely: the number of hidden layers, and the number of neurons in the hidden layers. Table 14 shows the results of the RMS of the

error and the R^2 of the model for the different functions depending on the architectural configurations (indicated by the number of neurons per layer).

Regarding the number of hidden layers, previous research shows that one is usually sufficient to model most of non-linearly separable problems but higher numbers can yield some performance gains when modeling problems with discontinuities such as a saw tooth wave pattern (Nelson 1994). Increasing the number of hidden layers can improve the model, but it can also introduce a greater risk of converging to local minima when dealing with highly nonlinear functions.

Table 14: Influence of the neural network architecture on the performance of the predictive decision support tool

Function	Neurons	RMS Error	R^2
1	1	44,812	0,7021
	10	7,650	0,9922
	100	8,867	0,9895
	100;10	5,387	0,9961
	100;50;10	1,054	0,9999
2	1	7001,1	0,7325
	10	2231,0	0,9645
	100	1498,6	0,9897
	100;10	942,5	0,9959
	100;50;10	332,5	0,9995
3	1	847532,0	0,7892
	10	178941,3	0,9915
	100	203648,5	0,9885
	100;10	424807,3	0,9226
	100;50;10	40151,5	0,9996
4	1	0,467	0,8640
	10	0,286	0,9493
	100	0,354	0,9213
	100;10	0,324	0,9348
	100;50;10	0,205	0,9739
5	1	7,119	0,7129
	10	0,967	0,9953
	100	1,323	0,9913
	100;10	0,424	0,9991
	100;50;10	0,266	0,9996
6	1	0,856	0,7233
	10	0,178	0,9900
	100	0,235	0,9820
	100;10	0,117	0,9957
	100;50;10	0,029	0,9997
7	1	26,961	0,0924
	10	27,785	0,0369
	100	27,681	0,0440
	100;10	23,997	0,2815
	100;50;10	20,046	0,4987
8	1	0,344	0,6206
	10	0,143	0,9392
	100	0,341	0,6509
	100;10	0,243	0,8264
	100;50;10	0,098	0,9718
9	1	19,484	0,0750
	10	19,692	0,0557
	100	19,771	0,0479
	100;10	19,593	0,0644
	100;50;10	6,943	0,8822

There were considerable gains in considering an hidden layer in the performance tests conducted, with the type of functions introduced in Table 13. For all the functions, the lowest RMS error and the highest R^2 values were obtained for the configuration with a hidden layer. This evidence was a result of the training vs. testing error divergence detection rule implemented (§5.3.2), which avoided the «bias/variance dilemma» and minimized the risk of converging towards local minima.

Another important parameter of the performance of the multilayer perceptron model is the number of neurons per hidden layer(s). If the number of neurons is insufficient, the network may not be able to accurately model the complexity of the data, and therefore perform poorly. If the number of neurons is too high, the network may over fit the training data. The most widely used heuristic guidelines to determine the number of neurons per layer are only tailored to one hidden layer configurations (Lai et al. 2005) and recommend that the number of neurons in this layer should follow one of the following rules (Nelson 1994):

$$n_{hid} = \begin{cases} \dfrac{n_{in} + n_{out}}{2} \\ \sqrt{n_{in} n_{out}} \\ n_{in} + n_{out} \\ 2(n_{in} + n_{out}) \end{cases}$$

Where n_{hid} is the number of neurons in the hidden layer, $n_{in}$the number of input neurons, and n_{out} the number of output neurons.

The result of the performance tests presented in Table 14 verify the initial expectations, when looking at the case with a single hidden layer and increasing the number of neurons in this layer, for most of the functions (1, 3-8, 9) the performance of the model is more accurate with 10 neurons than with 100 (showed both in lower RMS error and higher R^2 values). This is evidence of over-fitting the training data, for these functions the optimum number of neurons would be below 100 neurons.

For the performance tests described in sections 0 to 5.5.4, the architectural configuration used was the one with the best performance in this test, i.e. 100 neurons in the first, 50 in the second, and 10 in third hidden layer.

5.5.2. Influence of the number of samples on the dataset

One of the common criticisms of ANN models is that they require a training dataset with a high number of samples, the accuracy of the model is highly dependent on the number of samples in the training dataset. Indeed, the performance tests conducted with the earlier functions, for the number of samples varying between 10 and 500 (Table 15), showed that the quality of the solution provided by the model increased with the number of samples in the training dataset for most of the functions (a few of them had some insignificant loss of performance). There was a significant increase in terms of accuracy for all the functions tested (lower RMS error value and higher R^2) when the number of samples passed from 10 to 100, especially when looking at the most complex functions. When the number of samples increased from 100 to 500, the performance gain was, in most cases, marginal.

It is nevertheless important to understand that increasing the number of samples might not increase the accuracy of the prediction if the samples are not in the same order of magnitude as the desired range of prediction, i.e. if the designer intends to obtain a prediction in the range [0-10], it might be better to filter the dataset and train it with 100 samples within this range than with 1000 samples in the range [0,100].

Table 15: Influence of the number of the samples in the dataset on the performance of the predictive decision support tool

Function	Number samples	RMS Error	R^2
1	10	5,012	0,9945
	100	1,475	0,9996
	500	1,054	0,9998
2	10	1232,8	0,9914
	100	298,4	0,9996
	500	332,5	0,9995
3	10	146022,6	0,9933
	100	28851,1	0,9998
	500	40151,5	0,9998
4	10	0,110	0,9746
	100	0,055	0,9949
	500	0,205	0,9987
5	10	2,740	0,8728
	100	0,248	0,9990
	500	0,266	0,9989
6	10	0,148	0,9922
	100	0,027	0,9998
	500	0,029	0,9997
7	10	27,972	-0,0682
	100	28,829	-0,0879
	500	20,046	0,4806
8	10	0,113	0,9592
	100	0,069	0,9815
	500	0,098	0,9914
9	10	18,125	0,2126
	100	6,980	0,8841
	500	6,943	0,8853

For the performance tests described in the sections 5.5.1, 0 and 5.5.4, the architectural configuration used was the one with the best performance in this test, i.e. 500 data samples.

5.5.3. Influence of the number of trials

The initial choice of the samples that are used to train the network (update weights) and the samples that are used to test its results is random, therefore, it can be expected that a higher number of trials will lead to better results. In general, by increasing the number of trials, the combinations of training and testing data samples arrangements increases, and some of the prediction errors can average out.

This tendency was clearly verified by the results of the performance tests presented in Table 16, the accuracy of the predictions increased with the number of trials for most of the functions (with the exception of function 7 which is periodic). In some of the cases, the results obtained with a single trial were better than the ones with 5 trials, due to a non-representative fortuitous arrangement of the training set on the single trial that was analyzed.

Table 16: Influence of the number of the trials on the performance of the predictive decision support tool

Function	Number trials	RMS Error	R^2
1	1	0,939	0,9999
	5	1,361	0,9998
	10	1,055	0,9999
2	1	464,3	0,9990
	5	539,6	0,9987
	10	332,8	0,9995
3	1	115122,0	0,9965
	5	82380,0	0,9982
	10	40191,7	0,9996
4	1	0,338	0,9265
	5	0,234	0,9661
	10	0,205	0,9739
5	1	0,349	0,9994
	5	2,935	0,9996
	10	0,266	0,9997
6	1	0,065	0,9987
	5	0,030	0,9997
	10	0,029	0,9997
7	1	6,285	0,9490
	5	18,948	0,5520
	10	20,066	0,4987
8	1	0,282	0,7670
	5	0,028	0,8977
	10	0,098	0,9718
9	1	19,878	0,0378
	5	12,774	0,6031
	10	6,950	0,8822

For the performance tests described in the sections 5.5.1, 0 and 5.5.4, the architectural configuration used was the one with best performance in this test, i.e. 10 trials.

5.5.4. Influence of dataset distribution and uncertainty

The last performance tests focused on determining the influence of both the dataset distribution and its uncertainty on the quality of the ANN model. The accuracy of the prediction is highly dependent on the quality of the training dataset, if most of the data samples are in the proximity of the desired prediction, the quality of the solution will be higher.

The data samples were arranged from 0 to 100. In order to test the effect of sample distribution, three datasets were generated with 500 samples, one centered in the lower values (normal distribution with a mean of 20 and a standard deviation of 30), one centered in the higher values (normal distribution with a mean of 80 and a standard deviation of 30), and an uniform distribution (with a mean of 50).

Table 17: Influence of the dataset distribution on the performance of the predictive decision support tool

Function	Mean	STD	RMS Error	R^2
1	20	30	2,904	0,9988
	50	-	1,054	0,9999
	80	30	10,991	0,9517
2	20	30	1256,9	0,9926
	50	-	332,5	0,9995
	80	30	346,3	0,9994
3	20	30	193863	0,9900
	50	-	40151	0,9996
	80	30	57853	0,9990
4	20	30	0,259	0,9558
	50	-	0,205	0,9745
	80	30	0,269	0,9435
5	20	30	0,200	0,9998
	50	-	0,266	0,9996
	80	30	0,754	0,9972
6	20	30	0,172	0,9901
	50	-	0,029	0,9997
	80	30	0,053	0,9990
7	20	30	17,538	0,6117
	50	-	20,046	0,4984
	80	30	19,211	0,5391
8	20	30	0,029	0,9674
	50	-	0,098	0,9717
	80	30	0,192	0,8911
9	20	30	8,755	0,8107
	50	-	12,201	0,6371
	80	30	20,537	0,0432

The verification values were uniformly distributed in the range from 0 to 100 and, as expected, the error was lower for the uniform distribution of the dataset samples than the other two (with the exception of the periodic functions).

The decision support tool was designed to be used during the conceptual design phase, during which a lot of assumptions are made that increase the uncertainty in the samples of the dataset which can compromise the quality of the decision.

The results in Table 18, Table 19, and Table 20 show the accuracy of the prediction decision support tool for different datasets with uncertainty of 10%, 30% and 50%, respectively. The uncertainty was added as an error into the different functions as a percentage of the range of the function.

$$Error = 100 * Uncertainty(\%) * [Max(f) - Min(f)]$$

Where $Max(f)$and $Min(f)$ are the maximum and minimum of the functions, respectively.

Table 18: Influence of the dataset distribution on the performance of the predictive decision support tool with an uncertainty of 10%

Function	Mean	STD	RMS Error	R^2
1	20	30	6,200	0,9944
	50	-	1,995	0,9995
	80	30	2,813	0,9988
2	20	30	765,3	0,9973
	50	-	337,7	0,9995
	80	30	511,8	0,9987
3	20	30	108164	0,9969
	50	-	51936	0,9993
	80	30	34066	0,9997
4	20	30	0,243	0,9633
	50	-	0,279	0,9366
	80	30	0,212	0,9592
5	20	30	1,349	0,9850
	50	-	0,309	0,9994
	80	30	3,392	0,9353
6	20	30	0,112	0,9960
	50	-	0,051	0,9992
	80	30	0,062	0,9988
7	20	30	17,219	0,6299
	50	-	14,013	0,7549
	80	30	14,946	0,7201
8	20	30	0,023	0,9984
	50	-	0,096	0,9731
	80	30	0,200	0,8811
9	20	30	8,639	0,8150
	50	-	15,558	0,4100
	80	30	20,209	0,0040

The results showed, as expected, that the accuracy of the prediction decreased when the uncertainty in the dataset increased, seen both in higher values of the RMS error and lower values of R^2 for all the functions except functions 7 and 9 (the periodic ones).

Table 19: Influence of the dataset distribution on the performance of the predictive decision support tool with an uncertainty of 30%

Function	Mean	STD	RMS Error	R^2
1	20	30	5,529	0,9958
	50	-	3,308	0,9979
	80	30	8,359	0,9902
2	20	30	1149,4	0,9940
	50	-	630,6	0,9981
	80	30	717,6	0,9975
3	20	30	225249	0,9866
	50	-	93401	0,9976
	80	30	106131	0,9966
4	20	30	0,159	0,9829
	50	-	0,210	0,9331
	80	30	0,160	0,9835
5	20	30	0,305	0,9993
	50	-	0,489	0,9987
	80	30	2,157	0,9713
6	20	30	0,366	0,9074
	50	-	0,049	0,9992
	80	30	0,050	0,9992
7	20	30	20,327	0,4763
	50	-	19,828	0,5092
	80	30	14,121	0,7498
8	20	30	0,026	0,9980
	50	-	0,125	0,9445
	80	30	0,160	0,9254
9	20	30	8,668	0,8148
	50	-	17,795	0,2284
	80	30	19,862	0,0363

Table 20: Influence of the dataset distribution on the performance of the predictive decision support tool with an uncertainty of 50%

Function	Mean	STD	RMS Error	R^2
1	20	30	7,555	0,9912
	50	-	7,010	0,9921
	80	30	10,696	0,9846
2	20	30	697,8	0,9977
	50	-	789,5	0,9966
	80	30	503,9	0,9988
3	20	30	147727	0,9942
	50	-	83349	0,9981
	80	30	40843	0,9983
4	20	30	0,201	0,9745
	50	-	0,213	0,9725
	80	30	0,150	0,9851
5	20	30	0,690	0,9972
	50	-	0,657	0,9978
	80	30	1,389	0,9852
6	20	30	0,133	0,9928
	50	-	0,081	0,9971
	80	30	0,064	0,9986
7	20	30	16,212	0,6675
	50	-	21,355	0,4305
	80	30	18,977	0,5478
8	20	30	0,025	0,9980
	50	-	0,079	0,9816
	80	30	0,154	0,9307
9	20	30	12,707	0,6015
	50	-	16,207	0,3598
	80	30	20,435	0,0234

When looking at the combined effect of the data distribution and its uncertainty, it was noticed that the influence of the sample distribution decreased at higher levels of data uncertainty, i.e. with 30% or 50% uncertainty there was no significant loss in terms of accuracy by considering non-uniform samples distributions. The errors introduced by using the tool with samples outside the range of the desired prediction can be negligible when compared with the effects of using a dataset with a high level of uncertainty.

5.6. Case studies

In order to measure the impact of applying the developed decision support tool, a set of six different case studies were performed with the help of 36 third year undergraduate engineering students.

The goal of these case studies was not to simulate the "automatic design" of complex engineering systems but, instead, emulate possible low level decisions that need to be taken during the design process of this type of systems. These case studies were tailored to measure if a predictive tool (implementing a ANN model) can be used to enhance the quality of the decisions made about non-deterministic behaviors by non-experienced users (during the conceptual design stage). The different case studies focused on low level problems where designers typically follow an inductive approach, and are asked to look at previous data from similar contexts to extrapolate a decision for the context.

Prior to having contact with the tool, the students attained a fast one-hour course on neural networks and their prediction capabilities, and followed a small example application performed by the lecturer. They were then led to do, at two stages, three case studies with the help of the tool and another three case studies without it.

The students were randomly divided into two groups that attended separate sessions. The first group began with the three case studies using the tool and continued during the second session with the other three case studies without the tool. The second group did the opposite. The arrangement of the samples was such that no single student ever repeated the same case study. In order to study the influence of the order at which a specific case study was performed, the samples were organized in such a way that every case study was performed at least once in every possible order (i.e. there would be a student working on the case study first, another one working on it second, and another one working on it third).

The influence of the duration of the experiment was also measured, by determining different time intervals for the case studies. Every student was assigned a maximum duration at each session, one case study with a maximum duration of 20 minutes and two case studies with a maximum duration of 40 minutes. The way the different case studies were assigned aimed at an exploration of the experimentation space as close to a full factorial DOE as possible (i.e. cover all the combinations of case studies, different orders and durations = 3!*3=18) in order to study the influence of each parameter independently (Robbins 1952).

Appendix 3 includes an example of the assignment the students were asked to complete after each decision period with and without the use of the predictive decision support tool.

At each session, the students were individually assigned three case studies and were asked for their best suggestion within the time requirements (20 or 40 minutes) given some specific value inputs. They would only proceed to the next case study when the time limit was reached.

During the work on the case studies without the use of the decision support tool, the students were free to use any ad-hoc analysis (manual or supported by tools) they knew (e.g. polynomial interpolations, splines, spreadsheets, Matlab, etc.). When they were using the ANN tool, they had to load the given dataset, choose a filtering condition and define neural parameters such as: number of layers, number of neurons per layer and stopping conditions. Depending on the quality of the solution (measured through the training and testing error) they could then decide on proceeding to a reiteration with a different filtering condition and/or other neural network definitions.

Table 21: List of case studies used to test decision support tool

	Case Study	Database	Multivariate/ Classification/ Regression	Complexity
1	CO2 Emission vs. Mass	Car Emissions Database from EU 2008 (122 entries)	No/No/Yes	High
2	Mass to LEO vs. Launch Mass	Launchers Database (135 entries)	No/No/Yes	Medium
3	Math 1	MathDatabase1 (500 entries)	Yes/No/Yes	Low
4	Math 2	MathDatabase2 (100 entries)	No/No/Yes	Medium
5	Iris Classification problem	Iris Database (149 entries)	Yes/Yes/No	Low
6	Thrust Calculation	Thrust Values generated from formula (500 entries)	Yes/No/Yes	High

The selection of the case studies was undertaken to include, as much as possible, realistic examples of possible applications for conceptual design phase activities. Six case studies were selected to cover both single and multivariate problems and classification vs. regression goals. They were ranked by judiciously taking the model complexity into account (i.e. the number of parameters and the non-linearity of the problem) (Table 21).

The case studies 1, 2 and 5 were generated from existing databases of gaseous car emissions (Emissions 2009), a database of existing space launchers into LEO (Low Earth Orbit), and the iris classification problem (Anderson 1935), respectively. The students were given the database without a specific entry (chosen, for the purposes of this thesis, outside of the marginal set of values within the given dataset), for which they were asked for their best suggestions.

The case studies 3, 4 and 6 where generated with ad-hoc models. The case studies 3 and 4 have no physical correspondence and are described by the following models:

Case Study 3 (Math 1): $\left[if\left(\left[(x^2+y^2)^{\frac{1}{3}}>1=true\right]\ and\ [Boolean=true]\right), x+y, x-y\right]$, where x, y and $Boolean$ were variables randomly generated, if and and are logical conditions.

Case Study 4 (Math 2): $\frac{0.01x^3}{e^{\frac{x}{5}-1}}+\frac{0.04x^3(y-0.5)}{e^{\frac{3x}{5}-1}}$ where x covers the range [0, 100] and y is a random variable between 0 and 1.

Case Study 3 is a simple multivariate problem that intends to show the application of the methodology to discrete logical conditions. Case Study 4 intends to test the robustness of the model by adding an extra parameter (*y*) to represent data noise on a single variable model.

Finally, the case study 6 is a model of the maximum thrust for a liquid propulsion rocket nozzle, given the section area of the Throat (A_t), the pressure at the combustion chamber (P_c), the pressure at the exit nozzle section (P_e), the atmospheric pressure (P_a), and the section area at the exit of the nozzle (A_e). It is a highly nonlinear model described the following equation (Sutton 2010):

$$Thrust = A_t P_c\left[\sqrt{\left(\frac{2\gamma^2}{\gamma-1}\right)\left(\frac{2}{\gamma+1}^{\frac{\gamma+1}{\gamma-1}}\right)\left(1-\frac{P_e}{Pc}^{\frac{\gamma-1}{\gamma}}\right)}+\frac{P_e-P_a}{P_c}\frac{A_e}{A_t}\right]$$

Where γ is the adiabatic index (in this case, for the air, it was set at 1.4).

5.7. Results and discussion

Since this support tool was intended to be used by designers, typically not experts on ANN models, it was important to abstract the complexity of the neural network architecture definition from the decision process. However, the typical process leading to a design suggestion is far from linear (Seo et al. 2002). The tool offers a measure on both the training and testing error, and allows the possibility of performing several iterations to reduce the error output. With this objective in mind the user may perform different iterations to filter the data, change the stopping condition, change the number of hidden layers, change the number of neurons, change the number of runs, etc.

Correctly filtering the training data can significantly reduce the error by avoiding the inclusion of misleading data into the training process. Reducing the training sample with the application of a filter can help reduce the error of the analysis, e.g. when looking for values within the range [0,100], the user might think about filtering the values order > 1000 since they would induce additional error in the analysis. Nevertheless, if the dataset is over filtered, the number of samples in the dataset might be too small and induce high testing errors. Successive iterations can help find the point at which the dataset becomes over filtered. The user can search for the best distribution of the sample for its specific application by keeping a reasonable number of samples in the dataset that are representative for the orders of magnitude of the problem in question.

The choice in the number of hidden layers relates to the linear separability of the problem, an increase in the number of neurons per layer should typically increases the nonlinearity of the analysis, i.e. architectures with higher number of neurons can better approach nonlinear problems. Unfortunately, literature shows that sometimes increasing the number of neurons can lead to inadvertently model the noise of the training data set (Arbib 1995; Nelson 1994).

Once all the suggestions provided by the users were gathered, the suggestion error ($E_{Suggestion}$) was estimated by:

$$E_{Suggestion} = 100. \left| \frac{Suggested_{value} - Expected_{value}}{Expected_{value}} \right|$$

Where the $Suggested_{Value}$ was the suggestion provided by the students, and the $Expected_{value}$ the expected value for a specific case study. The end result was an error percentage, measuring how far the suggestion was from the correct value. An extremely good prediction would have an error of 0%. The results were filtered so that the rare suggestions with errors higher than 200% were considered as outliers and, therefore, not taken into account.

One of the advantages of the ANN tool approach is confirmed by the case studies: when using the tool the users always arrived to a suggestion, as opposed to a «no answer». This was not always the case when they were asked for suggestions without the use of the tool. This suggests that the use of such an approach can help detaching the quality of the design process from the personal skills of the designers. With accurate datasets and appropriate filtering process, non-expert designers can get meaningful suggestions without actually mastering the subject and/or the specific models that rule a certain design decision. The flexibility of the conceptual design process increases as it becomes less dependent on the availability of expert users.

An initial descriptive analysis showed that there was a significant difference in the error distribution when the users used the neural networks decision support tool, showing that the tool helped increasing the quality of the design suggestion for the six case studies with a direct impact on the quality of the design.

However, the initial descriptive analysis of the results also showed that the error distribution was far from a half-normal distribution (Table 22). The Kolmorov-Smirnov statistical test (Parsons & Wirsching 1982) showed virtually no correlation with normal distributions for all case studies except Case study 6 (where the 2-tailed $\alpha = 0.297$). Through a P-P Plot it can also be observed that the cumulative probability distribution was far from the Half-Normal (Figure 42).

Table 22: Normal and Half-Normal descriptive statistic parameters for the 6 case studies

Case Study		Normal distribution that best suits data					Half-Normal distribution that best suits data
		μ	σ	N (DOF+1)	K.S. Z (Kolmorov-Smirnov)	α (2- tailed)	Scale Factor
1	CO2 Emission	28.31	54.91	24	1.69	0.007	63.22
2	Launcher	31.75	52.60	24	1.96	0.001	58.87
3	Math 1	40.63	61.48	19	1.48	0.025	72.33
4	Math 2	20.11	27.33	26	1.36	0.050	33.51
5	Iris	7.93	26.67	27	2.63	0	27.35
6	Thrust	44.87	50.66	15	0.97	0.297	66.40

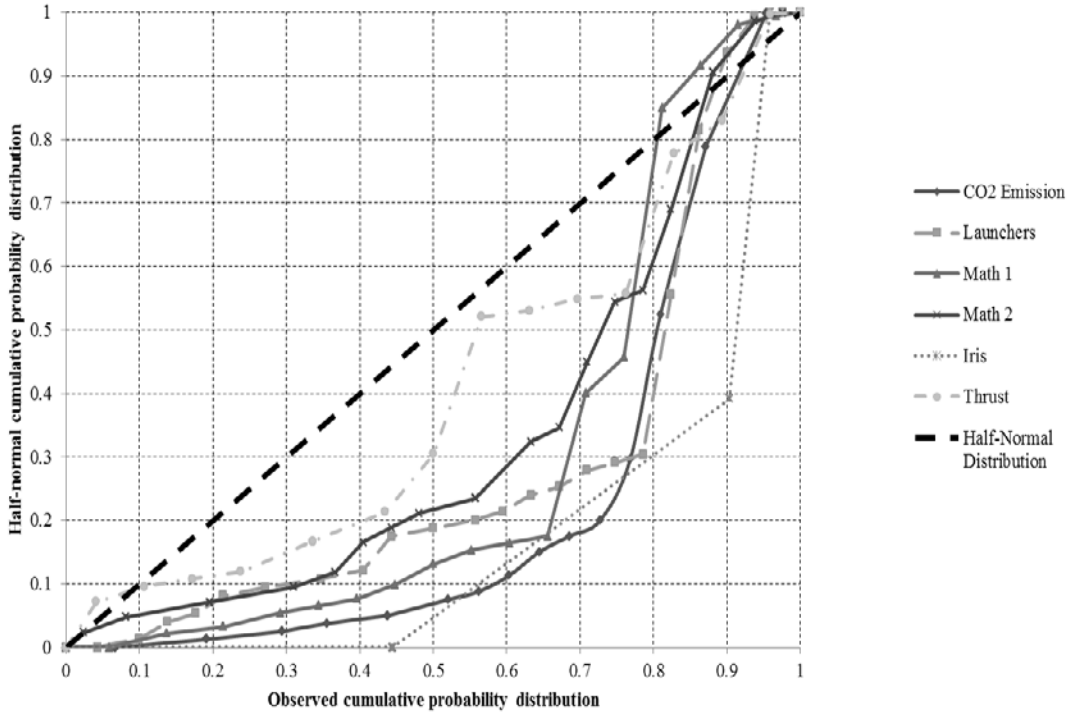


Figure 42: P-P plot where the observed cumulative probability distribution for each case study is plotted against the Half-Normal distribution

In addition to the mean and standard deviation, it was therefore necessary to consider the total distribution of the results. Figure 43 shows a box plot with the error distribution achieved for the different case studies with and without the use of the decision support tool (stars represent extreme points – distancing more than 3 times the interquartile range from the upper quartile, while circles represent outliers – distancing between 1.5 and 3 times the interquartile range from the upper quartile).

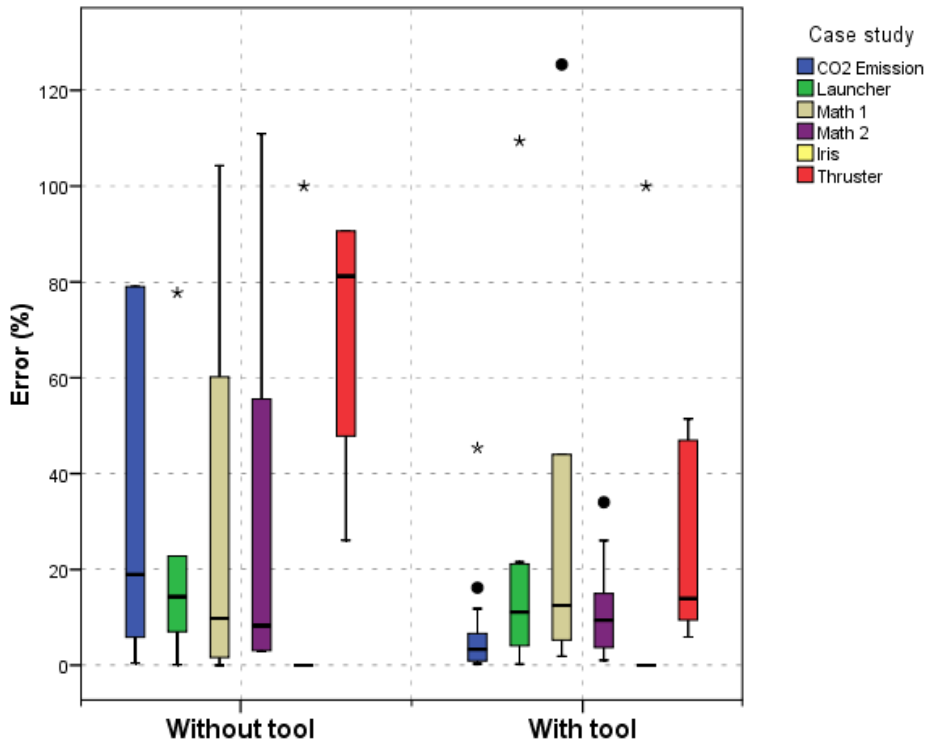


Figure 43: Design error for the 6 case studies with and without the decision support tool

In some of the case studies (1, 2, and 6), the advantage of using the decision support tool was clearly observed in the reduction of the median of the error. For the remaining case studies the impact of using the support tool cannot be directly observed from the median analysis, but is clear when analyzing the mean error results. If two error distributions have equal medians but different means, the one with lower mean value is the more proficient. Therefore, it is important to not only have a measure of the median of the error distribution but to also have a measure of the dispersion within the sample of users.

A possible figure of merit $(Gain(\%))$ encompassing both these concepts was used to distinguish the cases where the median of two distributions are the same but there is a considerable difference in the mean value.

$$Gain(\%) = 100\left(1 - \frac{\mu_{with}}{\mu_{without}}\right)$$

Where $\mu = (Median + Mean)/2$ is a measure of the aggregated impact of the support tool on the error distribution by considering both the mean and the median values in equal proportion, $\mu_{with}$and $\mu_{without}$ are the errors with and without the tool, respectively (

Table 23).

Table 23: Error distribution and gain when using decision support tool for each case study

Case Study		Use of tool	Error (%)				Gain %
			Count	Median	Mean	μ	
1	CO2 Emission	Without Tool	13	19	61	40	**87.5**
		With Tool	14	3	7	5	
2	Mass to LEO vs. Launch Mass	Without Tool	11	14	12	13	**30.8**
		With Tool	14	11	7	9	
3	Math1	Without Tool	9	10	36	23	**13.0**
		With Tool	10	12	28	20	
4	Math2	Without Tool	11	8	59	33.5	**19.4**
		With Tool	14	9	45	27	
5	Iris	Without Tool	11	0	13	6.5	**15.4**
		With Tool	15	0	11	5.5	
6	Thrust	Without Tool	6	81	89	85	**77.6**
		With Tool	9	14	24	19	

A considerable performance gain was obtained (in terms of quality of the design suggestion) when applying the tool in all the case studies. This can be observed both in the median and the mean of the error distribution. The gain also increased with the complexity of the problem, i.e. the more complex the case study, the higher the gain measured in terms of performance. This performance gain ranged from 13% for a low complexity mathematical model to 88% for a high complexity continuous case study.

Additionally, the users were asked to provide an assessment of the uncertainty they were expecting at each suggestion. It was found that the error was within the expected uncertainty for 94% of the cases when the decision support tool was used. This only happened for 46% of the cases when the tool was not used. This result shows that displaying both the training and the testing error of the analysis induces a higher perception of the error accuracy from the user's perspective.

It could be expected that the repeated use of the tool would lead to a better performance, showing the contribution of the learning factor. However, even though this tendency was verified at some samples, it was not a general trend. Non-parametric tests indicated that there was no considerable difference in the error distribution with the ordering of the case studies (σ = 0.166 at the Kruskal-Wallis test (Breslow 1970) for the three samples corresponding to the three possible orders). Further testing must be performed with other types of case studies to ascertain the influence of the learning factor on the quality of the prediction obtained with the decision support tool.

The duration of the case studies had almost no impact on the error distribution. There was no considerable difference in the error distribution by changing the duration of a case study from 20 to 40 minutes (2-tailed $\alpha = 0.996$ at the two sample Kolmorov-Smirnov test). This can be due to the relatively short intervals of time used, and should be more accurately perceived when dealing with more complex problems where the duration of the analysis is not completely constrained.

Once the case studies were finished, the users were asked to complete a small 10-question survey to assess their opinion about the different capabilities of the decision support tool, problems that they witnessed, and the usefulness of such a methodology for possible problems during the conceptual design phase (Table 24).

Table 24: Results of survey assessing opinion of users on the use of the methodology for possible conceptual design problems (result of 7 closed questions of the 10-question survey)

	Very easy and intuitive	Easy	It requires some effort	Difficult
Dataset definition	36 %	53%	11%	0%
Filtering process	25%	64%	11%	0%
Network architecture definition (layers and neurons)	8%	17%	58%	17%
Stopping conditions definition	16%	50%	25%	11%
Results visualization and exportation	17%	61%	22%	0%
Error measurement	31%	47%	14%	8%
Overall opinion	8%	56%	36%	0%

The results of the survey show that the users found some of the features implemented in the tool intuitive and easy to use (data set definition, filtering process, stopping conditions choice, results visualization, error information and exportation capabilities). The users also found that there were some activities requiring extra effort and/or previous specific knowledge: choosing the appropriate number of layers and neurons per layer. Through open-ended questions the users expressed the desire to have an even more automatic process, where the architectural definition of the neural network was automatically defined depending on the problem, e.g. when the dataset shows an almost linear model the tool

should reduce the number of layers and vice versa . The overall opinion also shows that they considered the tool is easy to use (64% = 56% + 8%), though 36% of the users were of the opinion that some effort is required.

Further work should focus on improving the feedback given to the designer, including advices on reiterations regarding the filtering process and/or neural network definition (e.g. the tool may advice the user to filter the training data within a certain range depending on the prediction that he/she wishes to obtain). A possible solution can be to guide the user in the selection process of the ANN parameters. Even though a feedforward multilayer perceptron architecture was chosen for its simplicity (Vellido, Lisboa, & Vaughan, 1999), other architectures can also be envisaged.

6. INTEGRATED CONCURRENT REAL-TIME ENVIRONMENT (INCREMENT)

The widespread presence of computers has allowed a change on the approach towards knowledge intensive activities ranging from basic document repositories to complex VM or KM strategies. This work focused on introducing, in Chapter 2, some of the numerous IT-enabled tools that can help increase productivity by helping to plan, establishing goals and modeling systems. However, when dealing with the conceptual design phase, regardless of the enhanced methods used, the primary activities will continue to be processing symbols and making decisions (Mistree et al. 1993), fundamental premises of the «human component». It is essential to develop tools that acknowledge this man-machine intricacy and, therefore, focus on maximizing the performance of both the design process and the baseline solution obtained without jeopardizing creativity, which is the most important added value activity introduced by the «human component».

The survey performed to experienced users detailed in Chapter 3 suggests that the current commonly used CSCD tools are not specifically tailored for the conceptual design phase and should be improved. A series of preliminary guidelines resulted in a formal proposal for the integrated design methodology introduced in Chapter 4, which focuses on fostering design collaboration by promoting a dynamic evolution of objectives, requirements/constraints, subsystems, system states, and design models, through four different stages with clear procedures.

The main concern of the engineering activities involved in system design is to predict the behavior of the physical phenomena typical of the system of interest. The development and utilization of mathematical models able to reproduce the future behavior of the system based on inputs, boundary conditions and constraints is of paramount importance for these design activities. The basic idea is that before those decisions that are hard to undo are made, the alternatives should be carefully assessed and discussed. Despite the favorable environment created by MBSE and Concurrent Engineering for the discipline experts to work, discuss and share knowledge, a certain lack of engineering-tool interoperability and standardized design methodologies has been so far a significant inhibitor (Estefan 2007).

The Integrated Concurrent Real-time EnvironMENT (INCREMENT) is a prototype software tool implementing some of the ideas behind the earlier integrated design

methodology. It supports some of the potential VM identified and KM strategies proposed, in an intuitive interface with a short learning curve.

Choosing a sample application to test the performance gain obtained with INCREMENT is an extremely challenging task. A simple case study is not able to clearly assess its main advantages. These can only be fully perceived in an environment with expert designers with a preliminary background/experience on the usual problems found during the conceptual design phase. «Systems thinking» is a prerequisite to maximize the potential performance gain of tools like INCREMENT, this concept cannot be learned without previously being exposed to situations requiring the design of complex systems (with emergent behavior) in a collaborative environment (CE or others).

Rather than an experiment with measurable results, this chapter introduce INCREMENT by presenting its main capabilities with the help of set of UML use case diagrams, and detailing its possible application to the design process of an Earth-observation satellite based on the subsystem models described on (Larson 1997) and (Fortescue 2003).

Since INCREMENT supports the methodology described in Chapter 4, the following subsections:

- briefly introduce the software architecture and graphical interface (§6.1). A more extensive description of all the functional capabilities, graphical user interface, and software architecture can be found in Appendix 4,
- indicate the design process setup activities (§6.2), and,
- detail a possible application of INCREMENT to support four different stages identified in the methodology of Chapter 4: general description (§6.3), requirements mapping (§6.4), interaction (§6.5), and optimization stages (§6.6).

6.1. Software architecture

INCREMENT's architecture was designed to support the conceptual design activities from the requirements' mapping stage to early design optimization. Rather than a repository, it is a design tool that enables and documents all the changes made to the system throughout the design process, including subsystems, design items, design parameters, assignment values, dependencies, parameter states, design items, model files, requirements, design decisions and justifications (Figure 44).

In order to reduce the required learning curve, the graphical interface is a ribbon on a common workbook editor (Figure 45). In the same platform, designers are able to define the subsystem level models, manage all the interactions, document every design change, and have a global view of the system.

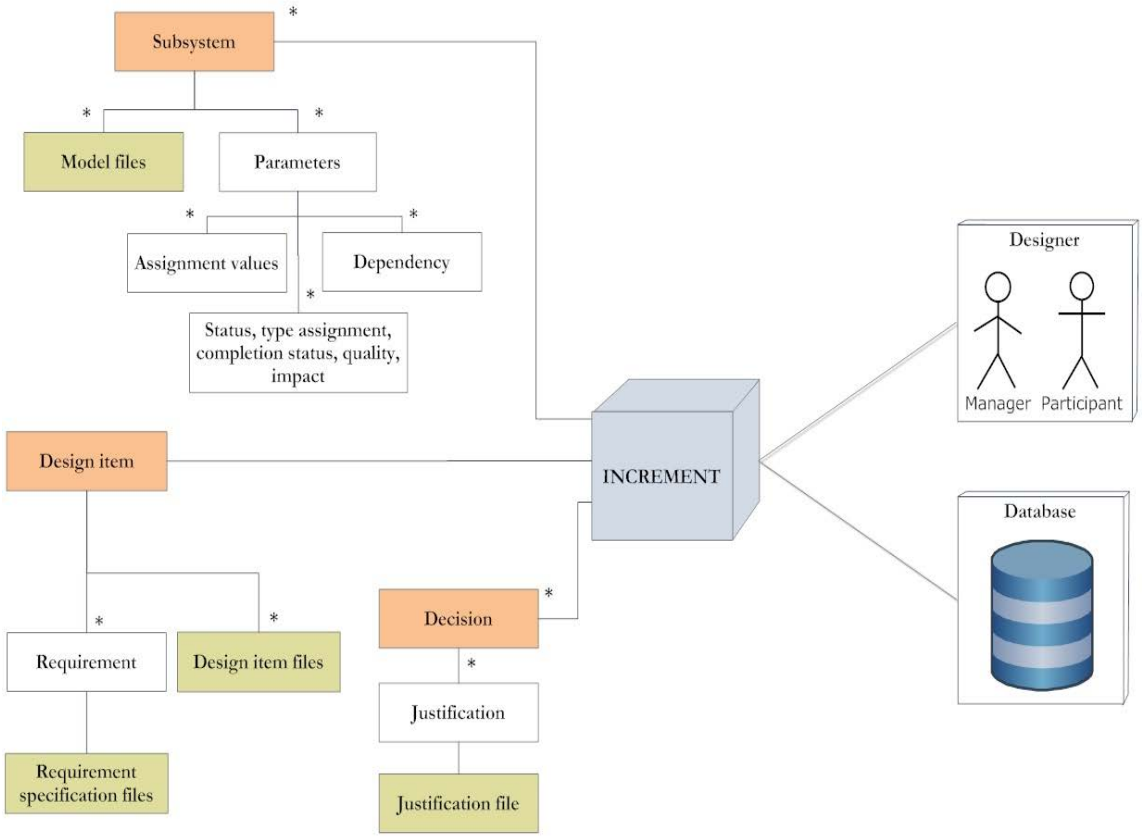


Figure 44: General architecture of INCREMENT

The graphical interface includes:

- a menu to establish a connection with the database and set up the interaction mode. The designer can use its login information to load all the current information formerly stored in the database. The designer can also define the type of interaction: synchronous (real-time) or asynchronous (not real-time) (§2.2) by launching the real-time update or not. These capabilities are detailed in §12.1.
- a manager menu, only accessible with manager credentials, that enables the definition of the different disciplines, participants, and permissions. It also allows the definition of the different states to monitor throughout the project with respect to: parameter assignment status, assignment types, assignment qualities, parameter

completion status, parameter impact, and system parameter groups. These capabilities are detailed in §12.2.

- a menu with participant functionalities. Allowing the definition of design items (such as design parts or system states), and mapping requirements/constraints into the system architecture. It is also in this menu that designers are able to change the attributes of the different design parameters that influence their specific subsystem(s). These capabilities are detailed in §12.3.
- a menu with visualization functionalities where the designer has access to the history evolution of the design parameters, design items or subsystem models. The designer can also select a «DSM view» of the system (detailed in §4.3), or a «dependencies view» to grasp precedencies or dependencies of specific design parameters, detailed in §12.4.
- a menu with the subsystem modeling functionalities allowing the incorporation of subsystem level models into the database through ordinary worksheets. These capabilities are detailed in §12.5.
- a menu enabling the display of the design possibilities (design results). Based on the assignments given to the input parameters of each subsystem, this option generates the design options that will be at the onset of tradespace exploration (or MDO) strategies. These capabilities are detailed in §12.6.

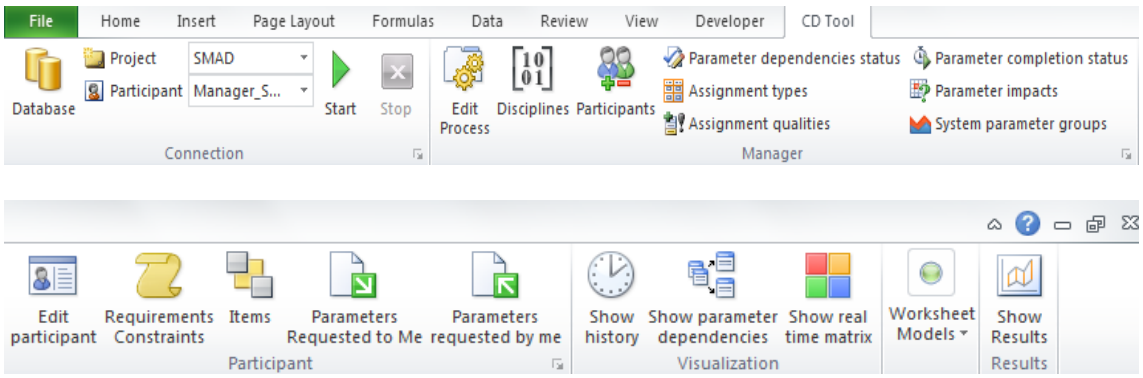


Figure 45: Graphical user interface of INCREMENT. Ribbon including the ability to connect to the system database, define the status of the parameters to keep track, map requirements, show design history, visualize the dynamic DSM, etc.

6.2. Design process setup

Before starting the design process it is important to establish the different roles in a typical concurrent design session. Table 25 details the typical roles taken by the different participants during the design process.

Table 25: Roles in a typical concurrent engineering design session

Role	Description
Manager	team member that represents the study manager
Team leader	team member that leads the concurrent design team
System engineer	team member that is responsible for system engineering and the overall system aspects
Assistant system engineer	team member that assists the system engineer
Domain/Subsystem expert	team member with particular skills in and knowledge of a specific domain, usually an engineering domain, and responsible for those aspects of a concurrent design study that relate to that domain
Observer	participant not actively involved in the study

To give an indication of team size, a typical concurrent design study team in the ESA CDF (ESA - CDF 2011) includes 1 manager, 1 team leader, 1 system engineer, 1 assistant system engineer, and approximately 18 domain experts, for a total of approximately 22 persons. In a number of studies the customer is represented by more than one person, e.g. a mission scientist / prime investigator.

The roles represent the different level of permissions given to each user within a specific system. Normally, the study manager and the team leader have access to all the capabilities (read and write), while the other roles have access to a smaller subset. As an example, the system engineer generally has access to all the subsystem/domains with the exception of the cost, and the domain/subsystem experts have typically only have write permission to its domain/subsystem and read permission to the upstream or downstream parameters that influence its design choices.

Another important notion is the definition of input domain/subsystem and output domain/subsystem for a parameter. The input domain/subsystem of a parameter, requests it because he/she assessed that it will have an impact on its subsystem level analyses. The output domain/subsystem of a parameter is requested for a meaningful value (or series of values) for a parameter by a certain input domain/subsystem, it should verify that the demand can be satisfied and, if so, provide an assessment of the requested value.

The use case diagram in Figure 46 lists all the possible actions that the study manager can perform to set up the design session. The other roles typically have more stringent constrains regarding these actions. The study manager is able to, for instance, create a new system, add/edit or delete participants (domain/subsystem experts), assign different

permissions to different participants, or change the design monitoring items that need to be monitored during the design phase. These design monitoring items relate to the five different categories defined in §4.3: parameter dependencies status, assignment types, assignment quality, parameter impact, and parameter completion status.

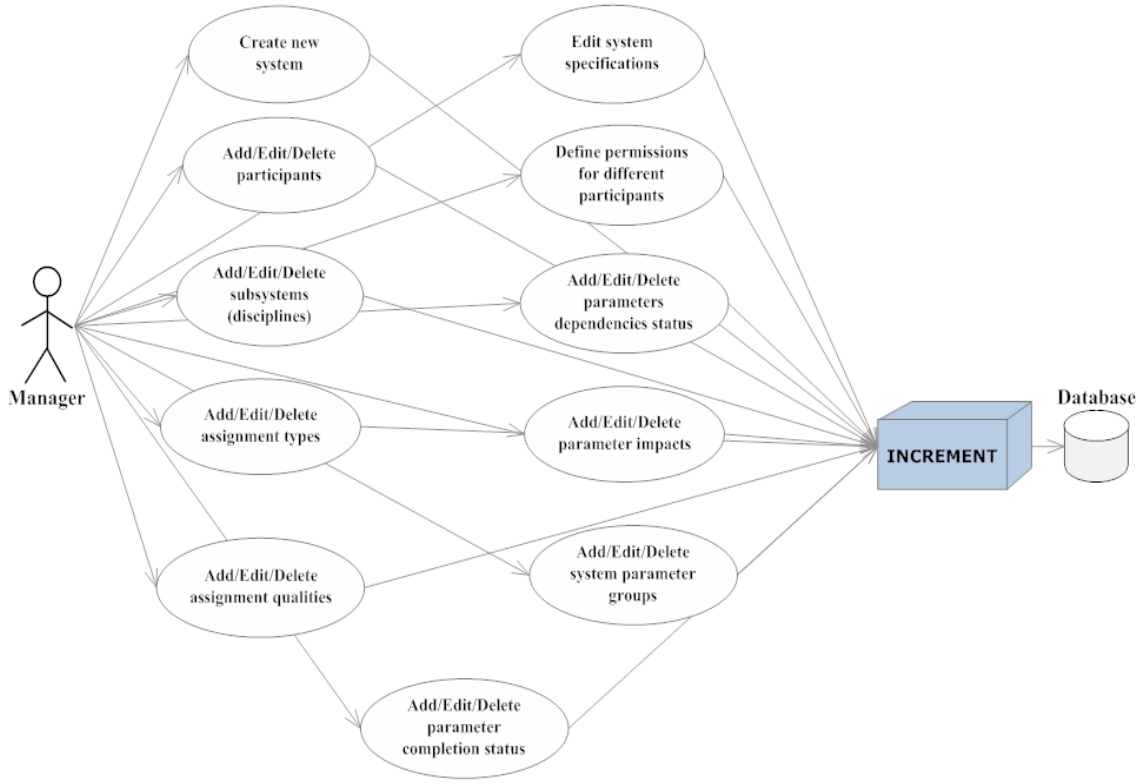


Figure 46: Use case diagram for the manager capabilities of INCREMENT during the design process setup

As an example, in the case of an Earth-observation satellite, the study manager may define that, in the case of the parameter dependencies status, it is important to keep track if a certain parameter has been *requested* by an input subsystem/domain (who asks for the parameter), *rejected* by the output subsystem/domain (which was asked for the parameter), *internal* (accepted but is not yet shared by the output subsystem/domain), *manual* (accepted and shared by the output subsystem/domain, but there is no underlying model defining its value), or *external* (accepted and shared by the output subsystem/domain and there is an underlying model defining its value).

These different parameter dependencies status, along with the ones defined by the manager for the other four categories (assignment types, assignment quality, parameter impact and parameter completion status), will afterwards be available to the all the domain/subsystem experts during the interaction phase to increase the insight on the design process.

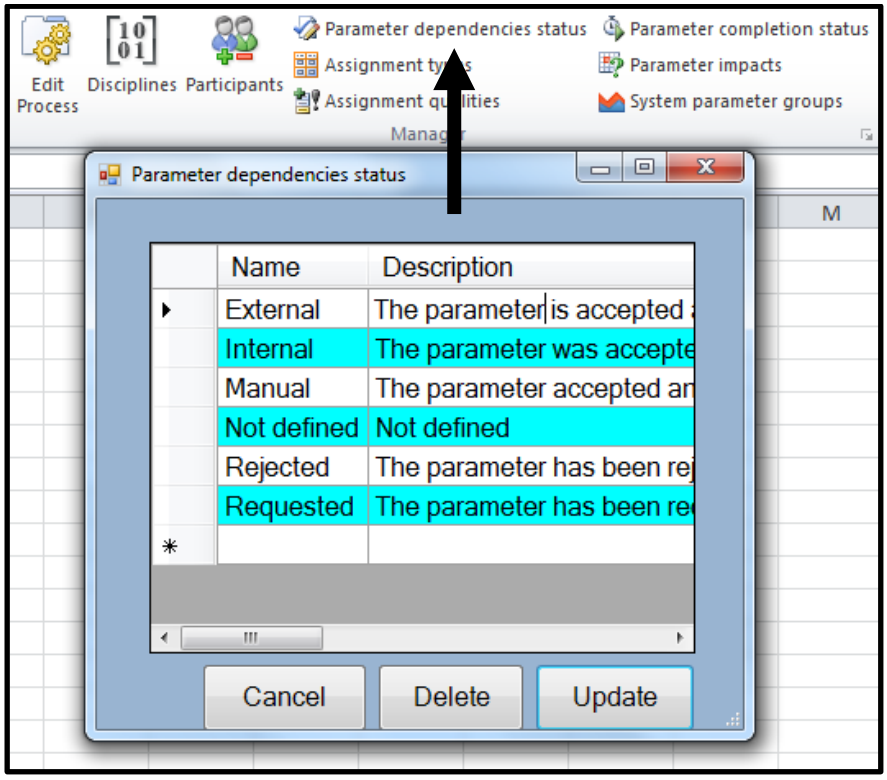


Figure 47: Example of the definition of the parameter dependencies status to monitor by the manager in INCREMENT. The different status will be available to all the domain/subsystem experts to classify their parameters during the interaction stage. In this example, the manager decide to monitor if a parameter is external, internal, manual, rejected, requested, or not defined as introduced in §4.3.1.

6.3. Supporting the general description stage

The methodology introduced in Chapter 4 defends that the concurrent design process begins with a high level objective and a list of requirements that trigger off the general description stage, which three lower levels goals:

- build a concept map around the system objective statement in order to clearly express what will be involved and is required to reach the objective,
- define the different system states corresponding to the different lifecycle phases of the system (e.g. launch, nominal mode, or eclipse in the case of a satellite), and,
- define the different types of physical and/or functional subsystems/domains corresponding to the main fields that will be considered for the latter design stages (e.g. structures or thermal in the case of a satellite).

As far as the concept map technologies are concerned, there are already numerous tools able to implement concept maps, some of the most prominent are (SmartDraw n.d.) or

(Cmap n.d.). Therefore, INCREMENT focused on enabling the creation of different system states and different subsystems/domains.

In the case of an Earth-observation satellite, the design process may start with an objective such as: "*Design an Earth observation mission to provide world-wide disaster-management capabilities, for over a period of 7 years*" and a list of high level requirements such as (Table 26):

Table 26: Example of simple requirements for the design of an Earth-observation satellite

Req. #	Title	Description
1	Data accessibility	The satellite shall revisit the same area on the Earth surface within 24 hours
2	Data management	The satellite and shall be able to send the acquired data back, in real time, to any equipped ground station (the reference ground station is considered with 1 m aperture antenna diameter) with a link margin of at least 4dB
3	Maximum payload mass	The maximum payload on polar orbit shall not exceed 2950 kg
4	Maximum cost	The total lifecycle cost of the mission shall not exceed million 350 million US Dollars (2012)
5	Revisit	The latest observation data shall be accessible to the national emergency authorities of all the European countries as an intuitive web-service agent

After the initial design setup activities described in the previous section (including the creation of a new system, definition of the different participants with their respective permissions, and the different design status to monitor), INCREMENT can be used to define the different lifecycle states that need to be taken into account during the conceptual design phase.

For example in the case of an Earth-observation satellite they can be: the *launch*, the *nominal* and the *eclipse* states (Figure 48).

The definition of the different system states will create be the basis for the future requirements mapping stage, where the requirements will be mapped to the different system states without providing early design hints on a specific architecture, or design solution.

Once the design states are identified the design process evolves to the definition of the different domains/subsystems. These correspond to the physical and/or functional domains, and are usually defined in a way that tries to minimize the interactions between subsystems (making a specific subsystem as hermetic as possible), by technical expertise, or other considerations.

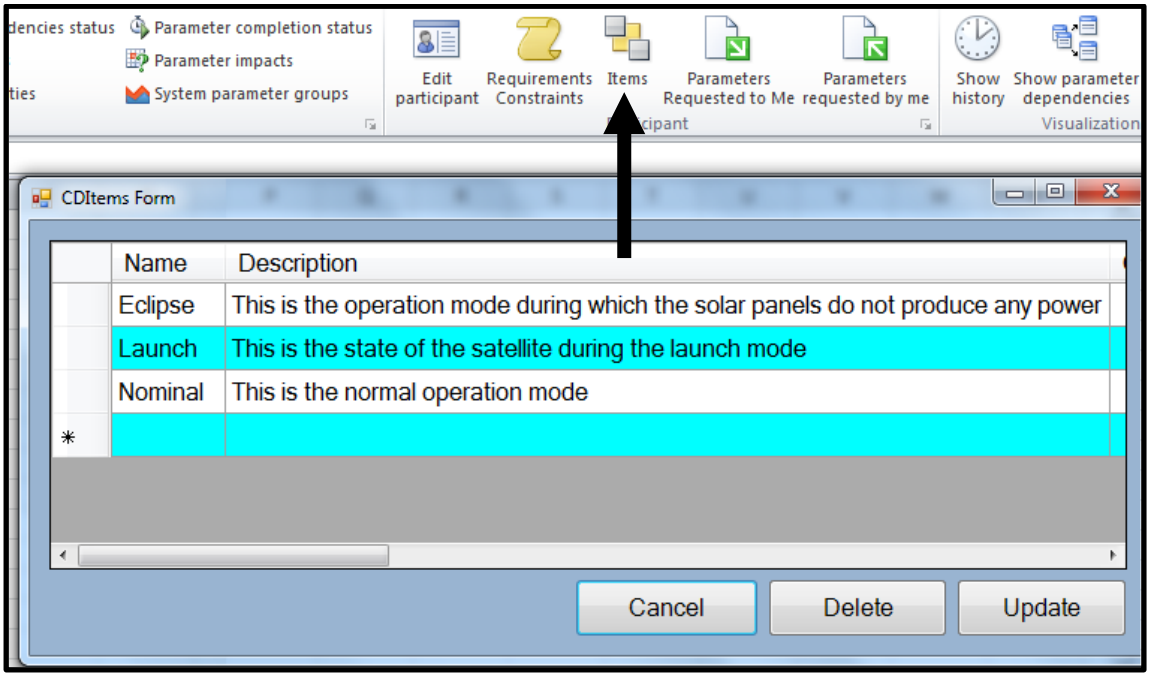


Figure 48: Definition of the different system states in INCREMENT

For example, in the case of an Earth-observation satellite these can be: Attitude Determination and Control System (ADCS), Communications, Propulsion, etc. (Figure 49).

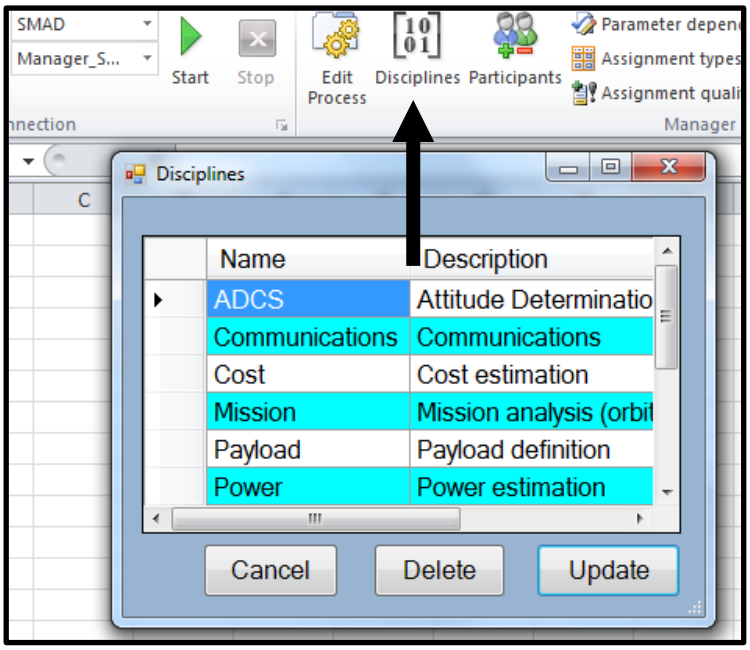


Figure 49: Definition of subsystems/domains in INCREMENT (example for an Earth-observation satellite)

6.4. Supporting the requirements' mapping stage

Once the general description stage is finished, the design process evolves to the requirements' mapping stage where the different requirements are mapped into the

different system states. The goal is to enable the designer to clearly know which requirements (or other constraints) have to be considered when considering a specific system state. It will also provide a way to support future design decisions that will emerge during the other phases of the design process.

INCREMENT supports several actions to manipulate requirements such as (Figure 50) : creation/edition/deletion of requirements, justification of requirements with text and/or external documents, and mapping existing requirements to previously defined system states (defined at the general description stage §6.3).

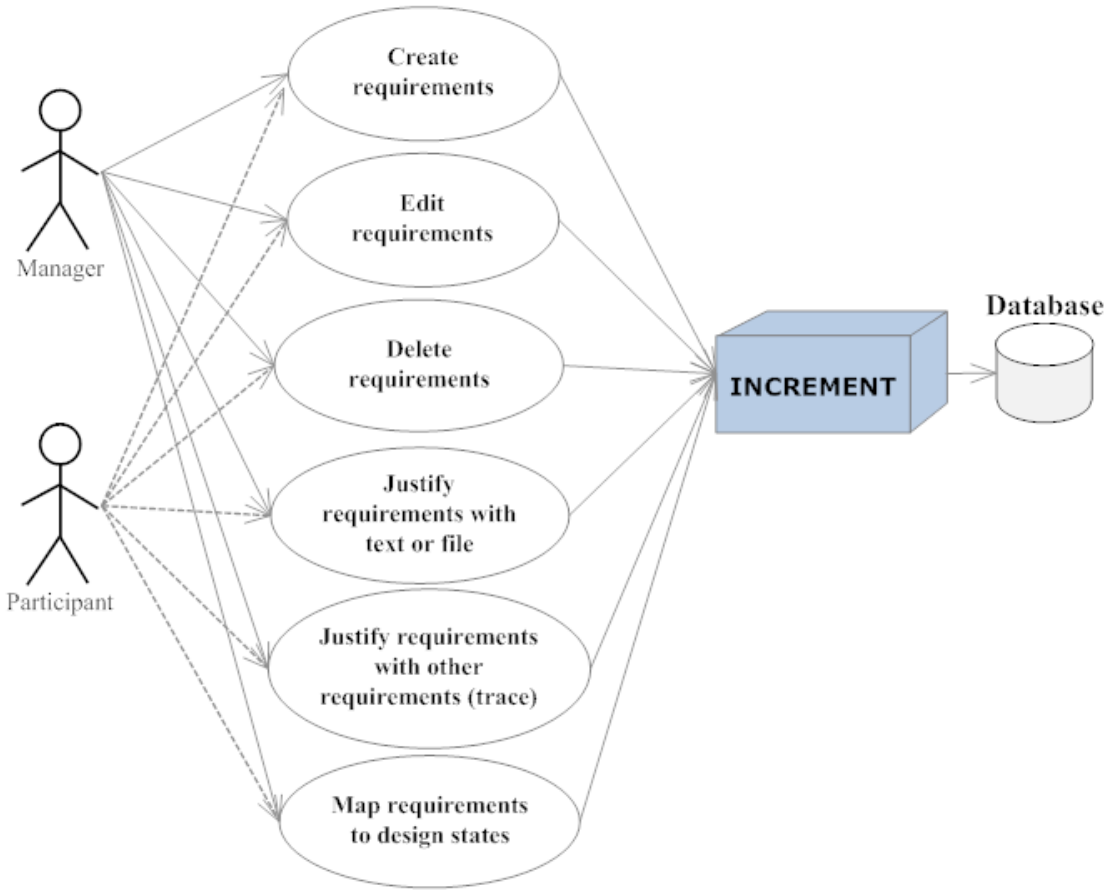


Figure 50: Use case diagram detailing the capabilities of INCREMENT to manipulate requirements

The participant is only allowed to access the requirements if he/she has read or write permission to the design states and/or subsystems to which the requirements are mapped. These rights are defined by the manager and can be changed throughout the design process.

Carrying on with the Earth-observation satellite, the different users of INCREMENT can create the initial system requirements (Table 26), e.g. data accessibility, data management, or maximum cost (Figure 51).

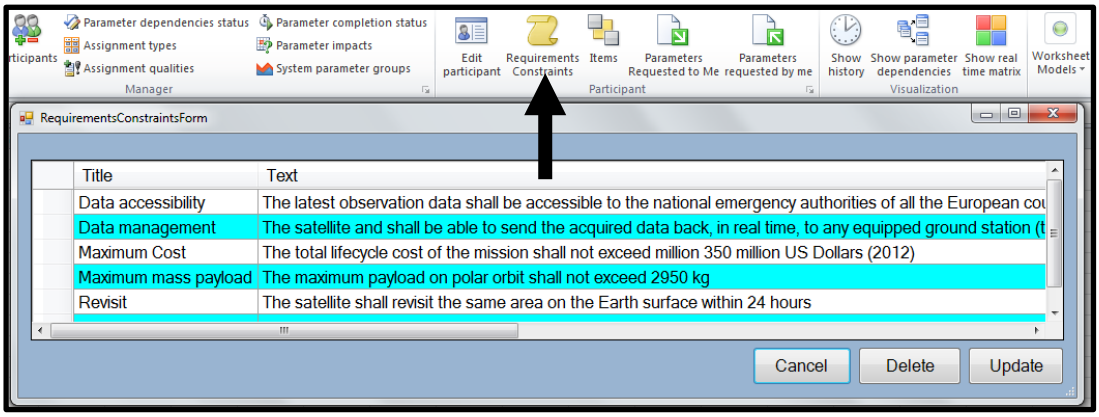


Figure 51: Example of the application of INCREMENT to model the initial requirements of the design process. Earth-observation satellite with five high-level requirements.

Once the different requirements are created, the requirements can be mapped to the different system states, e.g. the Maximum cost (Req. 4) and the Maximum mass payload (Req. 5) can be mapped to the eclipse state of an Earth-observation satellite (Figure 52).

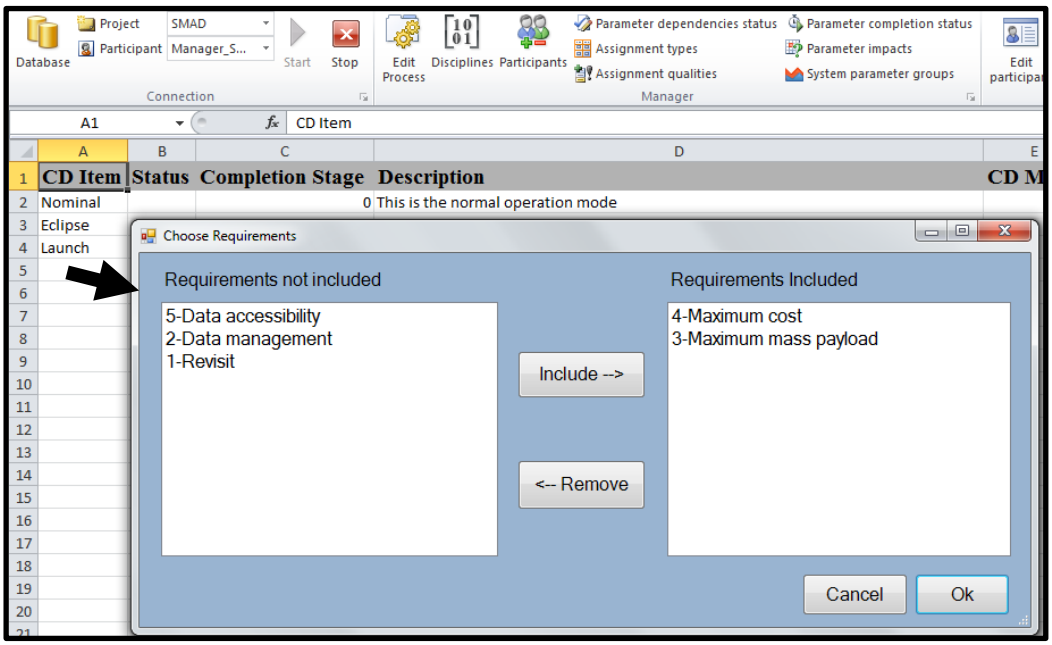


Figure 52: Example of the application of INCREMENT to map two requirements (Maximum cost and Maximum payload mass) to a system state (eclipse) of an Earth-observation satellite.

6.5. Supporting the interaction stage

This is the stage during which both parameter dependencies and assignments are changed and justified. It corresponds to the typical CE process where the designers responsible for

each subsystem work together to agree on the dependencies flow and perform design decisions (§4.3). It is at this stage that: new dependencies can be established by requesting parameters from other subsystems, existing dependencies can have their status changed, the possible values for a specific parameter can be changed, etc. Every alteration can be supported by requirements or other design justifications. INCREMENT supports all of these actions and others such as mapping grouping parameters for budgeting purposes e.g. mass or power budgets (Figure 53).

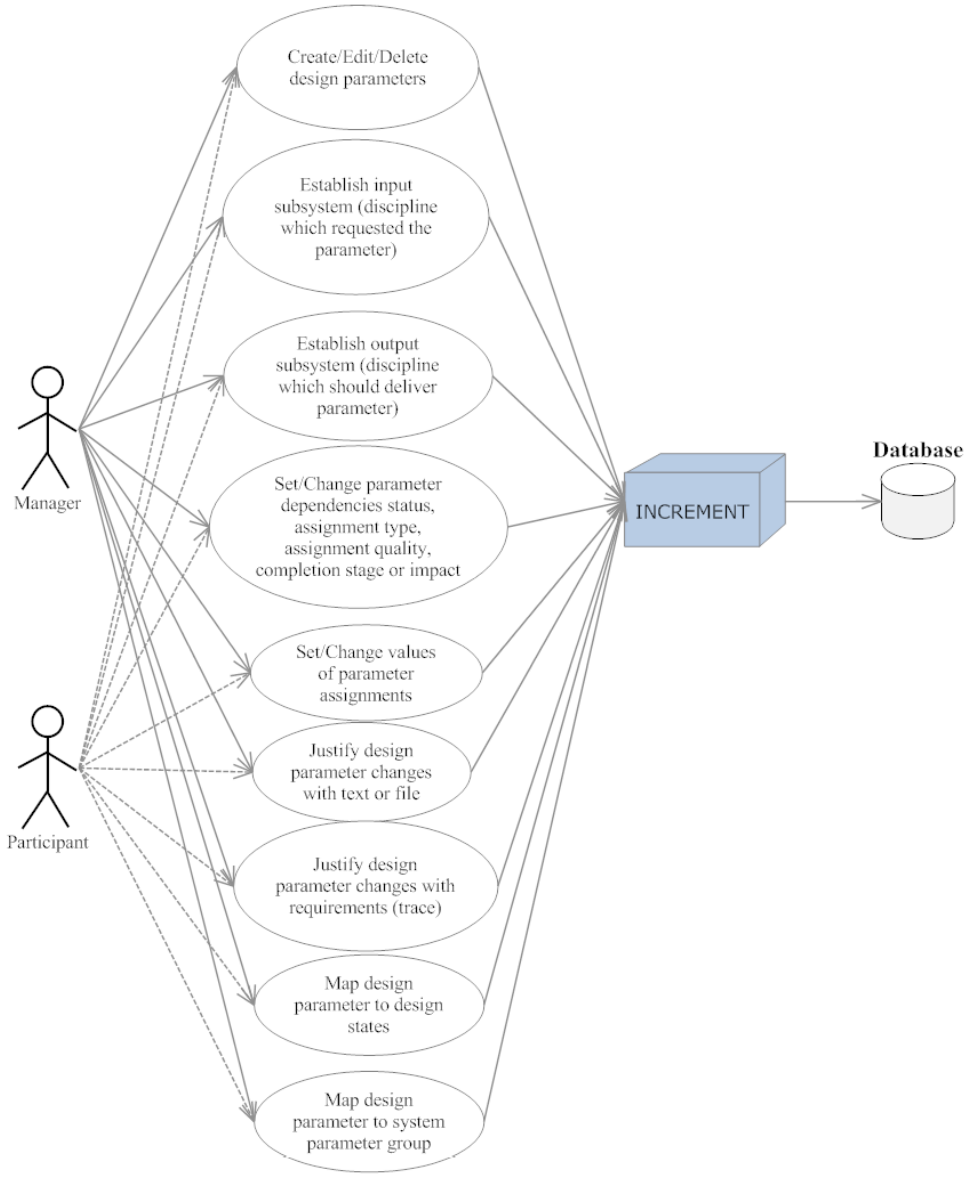


Figure 53: Use case diagram detailing the capabilities of INCREMENT during the interaction stage

In order to understand the support provided by INCREMENT at this design stage, it is useful to continue with the example of an Earth-observation satellite to show a usual design iteration involving different designers. In this simple example take part two

domain/subsystem experts (John – Mission analysis specialist, and Peter – Structural specialist) and a manager (which in this case takes the role of the System engineer) (Figure 54).

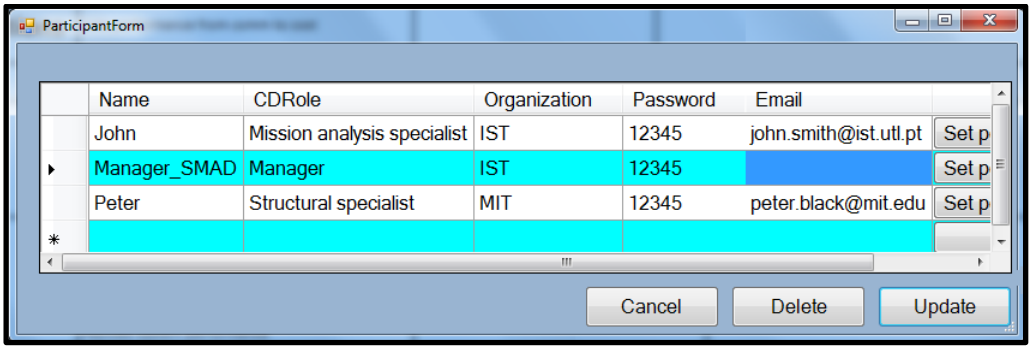


Figure 54: Different users in an example of a design iteration for an Earth-observation satellite

The example of design iteration has a total of **seven steps** to demonstrate parameter exchange intricacies, design justifications, and the use of the visualizations introduced in §4.3: dependency graphs and the dynamic DSM.

The changes in the dynamic DSM view will be monitored for the different steps according to the type of information detailed in §4.3.1 (parameter dependency status), §4.3.2 (parameter assignment type), §4.3.3 (parameter assignment quality), and §4.3.5 (parameter completion status). The visual metaphors follow the following schema:

Table 27: Visual metaphors for the dynamic DSM to use on a design iteration of an Earth- observation satellite

Type of information	Chosen metaphor	Different possibilities	Code
Parameter dependency status	Color	Requested	Red
		Internal	Yellow
		Manual	Blue
		External	Green
Parameter assignment type	Text	Single point	"(point)"
		Multiple	"(multiple)"
Parameter assignment quality	Pattern	Low quality	Vertical stripes
		High quality	Horizontal stripes
Parameter completion status	Text format	Problem	Underlined text
		Completed	Italic text

Step 1

Peter, the specialist of the "Structures" domain, requests an axial and a lateral load from the Mission analysis domain.

He sets the parameter dependency status to "Requested", which changes the color of the correspondent parameter in the dynamic DSM to red (Figure 55).

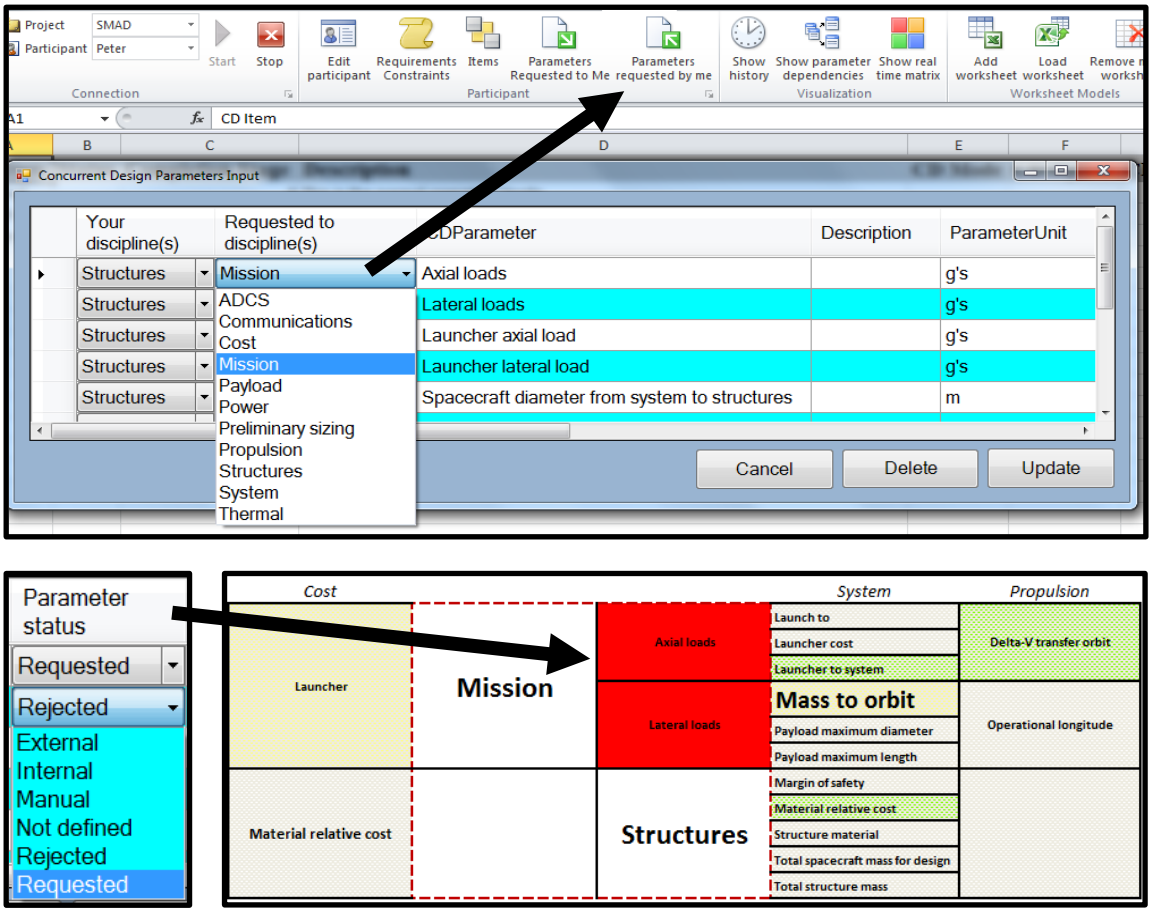


Figure 55: Example of the application of INCREMENT to monitor a parameter request. In this case the "Structures" domain requests the "Axial Load" and the "Lateral Load" parameters from the "Mission Analysis" domain. Since the visual metaphor associated with the "Requested" status is red, the dynamic DSM automatically changes to red on INCREMENT

Step 2

John, the specialist of the "Mission Analysis" domain, accepts the parameter requests, but still does not have an estimation of their value and therefore sets them to "Internal" to show that the parameters have been accepted but are not yet shared by the Output subsystem/domain (in this case the "Structures" domain).

Since the visual metaphor associated with the "Internal" parameter dependency status is the yellow color, the dynamic DSM changes automatically to yellow on INCREMENT (Figure 56).

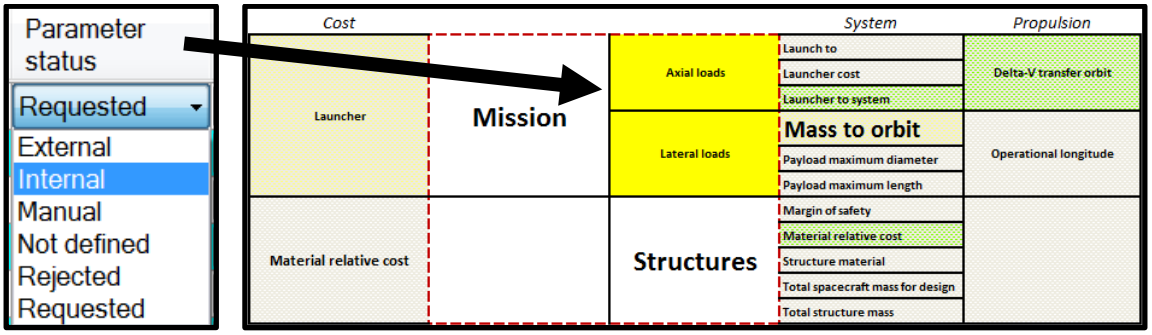
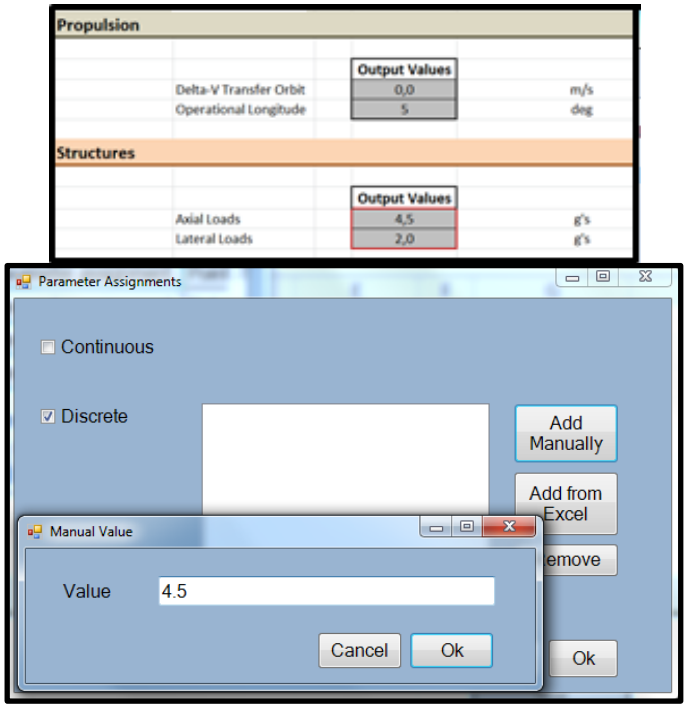


Figure 56: Example of the application of INCREMENT to change in the parameter dependency status. In this case the "Mission Analysis" domain sets the parameter dependency status to "Internal" to show it has been accepted but is not yet shared. Since the visual metaphor associated with the "Internal" status is yellow, the dynamic DSM automatically changes to yellow on INCREMENT

Step 3

After a certain time, John, the "Mission Analysis" specialist, provides his first guess on the value for both the Axial and the Lateral loads based on a previous mission with similar objectives. He makes a "point estimation" that the axial load will be of 4.5 g's and the lateral load of 2 g's.

Since he now has a value for the parameter, he sets the parameter dependency status to "manual", which is represented in the dynamic DSM by the blue color. Because it is only an initial guess, he sets the assignment quality to "Low quality", which is represented in the dynamic DSM with a vertical pattern. Finally, as it is a single "point estimation", he sets the parameter assignment type to "Point", which is represented in the dynamic DSM with the extra text next "single" to the name of the parameter (Figure 57).



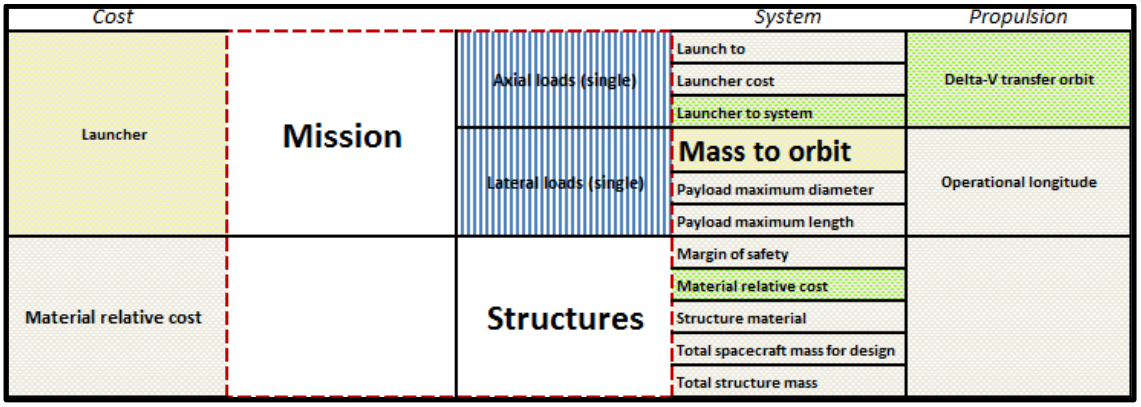


Figure 57: Example of the application of INCREMENT to change the parameter assignment value. In this case the "Mission Analysis" domain sets a "Manual" value for both the axial and lateral loads (represented with the blue color). Since it is only an initial guess, the parameter is set as "Low quality" (represented as a vertical pattern in the dynamic DSM), and a single point estimation (represented with the extra text "single" next to the name of the parameter in the dynamic DSM).

To support the changes made in step 3, John, the "Mission Analysis" specialist, provides a small justification text and traces it to an existing requirements which was mapped to the "Launch" state, in this case the "Maximum cost" requirement (Figure 58).

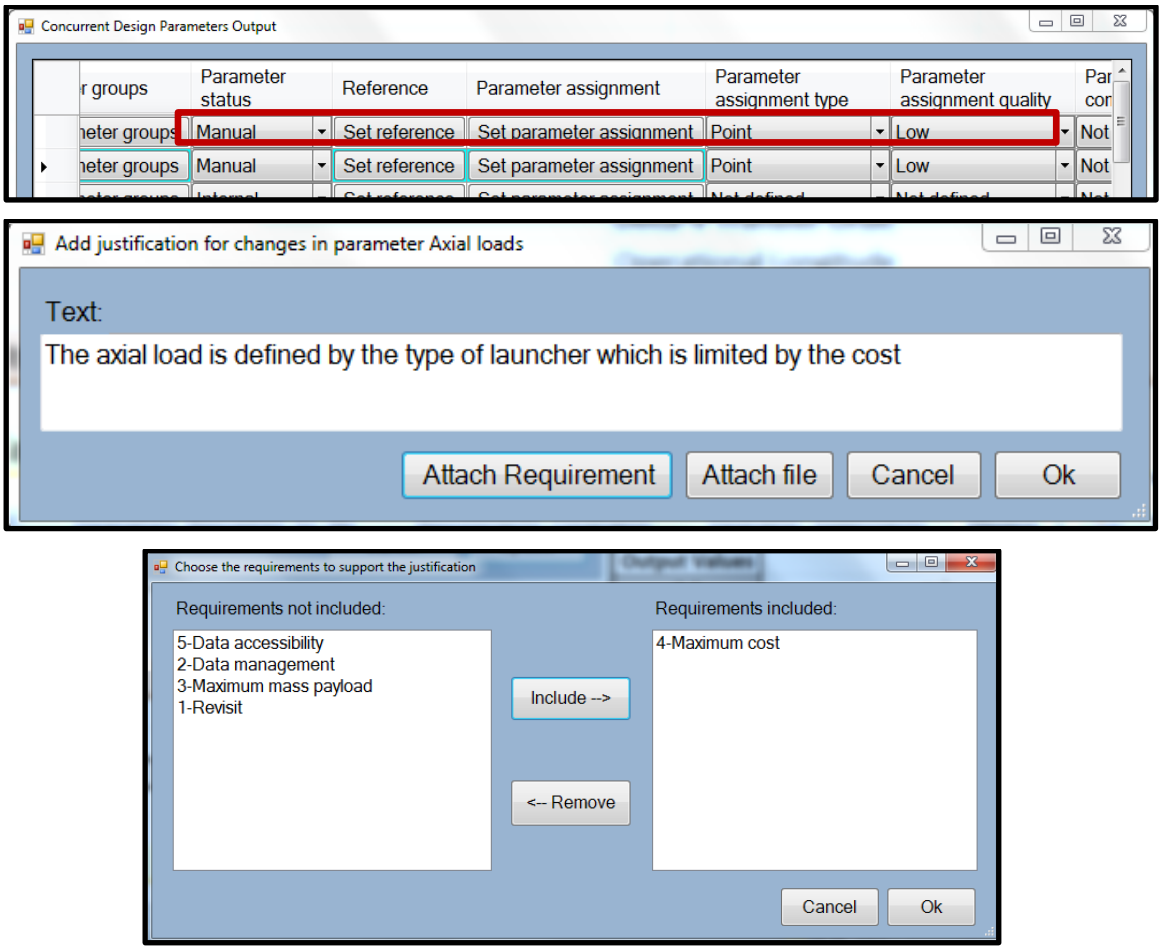


Figure 58: Example of the application of INCREMENT to justify a parameter value by tracing it to a previously defined requirement. In this case, the manual value given for both the Axial and the Lateral loads is justified with the choice of the launcher which, in its turn, is constrained by the "Maximum cost" requirement

Step 4

Once the "Structures" domain specialist, Peter, conducts its subsystem-level analysis, he realizes that the given values of Axial and Lateral loads result in a negative margin of safety in his domain/subsystem. Since this is not an acceptable situation, he decides to set the parameter completion status to "Problem" to show that there no agreement on the value of the parameter (amongst the subsystems/domains involved), requiring an intervention at the system level. This is shown by the underlined text on the name of the parameter (Figure 59).

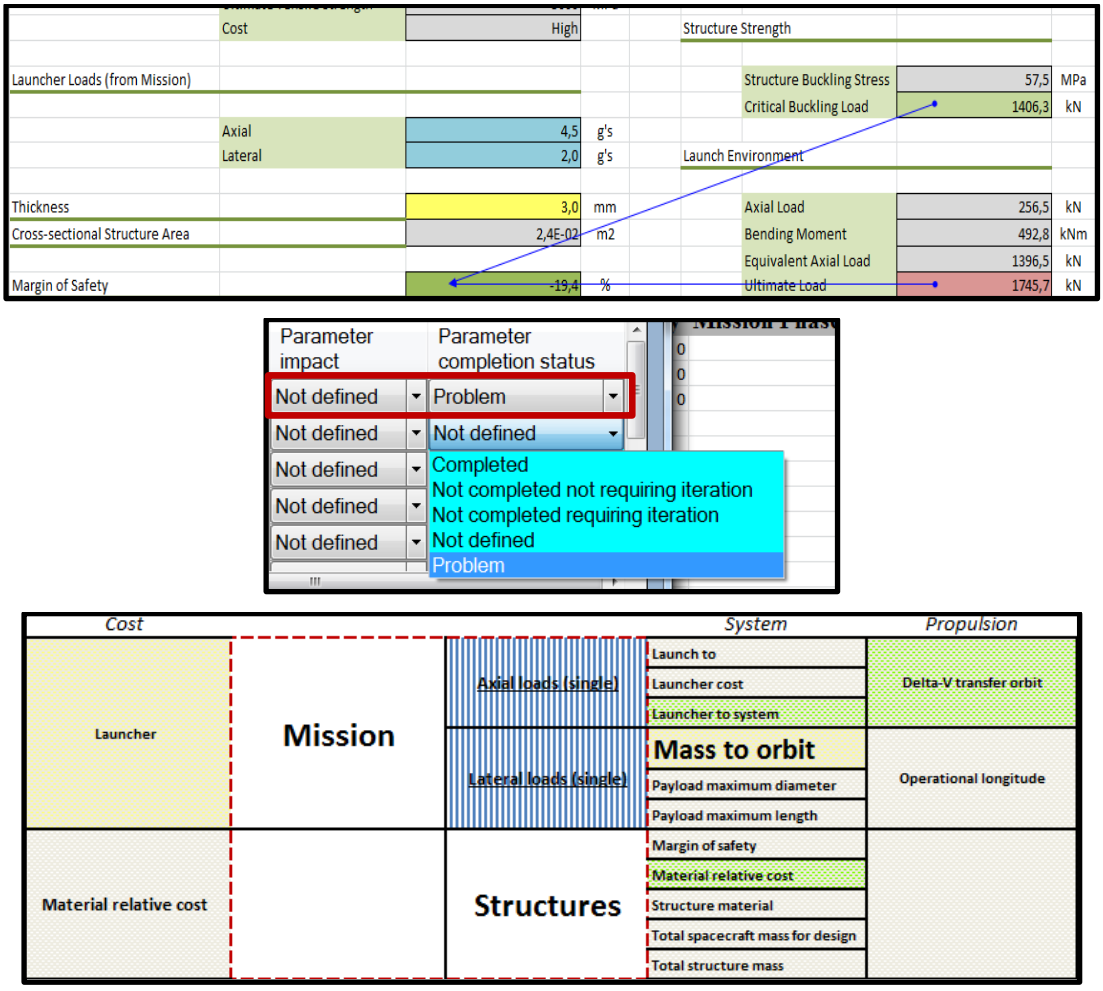


Figure 59: Example of the application of INCREMENT to change the parameter completion status to "Problem". Since the visual metaphor associated with the "Problem" completion station is the underlined text, the name of the Axial Load and Lateral parameters become underlined

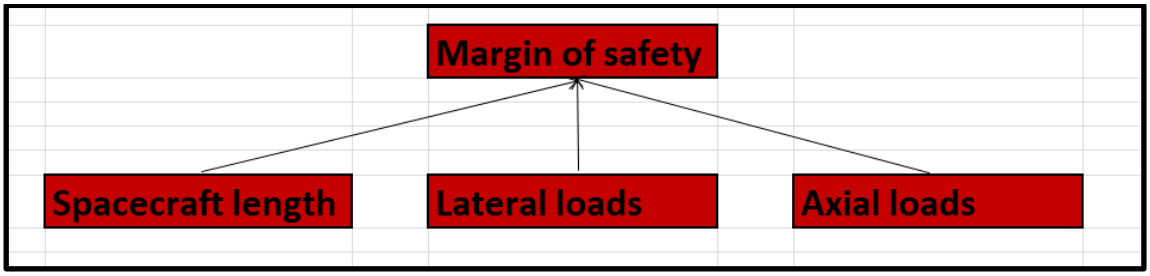
Step 5

The previous step required the intervention of the "system engineer". He/she decides to use INCREMENT to take a look at the dependency graph, and the historic parameter changes.

Looking at the dependency graph of the "Margin of Safety" he/she realizes that it depends on both the Lateral and Axial loads that were provided by the "Mission Analysis" domain.

He/she then takes a look at the parameter changes by looking at the historical changes made both at the Axial and the Lateral load parameters. He/she realizes that, once the parameter assignment value was provided manually (4.5 g's and 2 g's, for the axial and lateral loads, respectively), the parameter completion status changed into "Problem".

In order to sort out this blockage he/she asks John, the "Mission Analysis" specialist, to provide a more accurate estimation of these parameters (Figure 60).



Change Date	Parameter Name	Domain requesting	Domain requested	Parameter Assignment Value(s)	Parameter Assignment Status	Parameter Assignment Quality	Parameter Completion Status	Justification Text	Justification Requirement Title
27/2/2012 at 10:57:08	Axial loads	Structures	Mission	4,5	Manual	Low	Problem	-	-
27/2/2012 at 10:47:18	Axial loads	Structures	Mission	4,5	Manual	Low	Problem	Does not comply with the necessary safety margin	-
27/2/2012 at 10:37:55	Axial loads	Structures	Mission	4,5	Manual	Low	Not defined	The axial load is defined by the type of launcher which is limited by the cost	Maximum cost
27/2/2012 at 09:51:20	Axial loads	Structures	Mission	-	Internal	Not defined	Not defined	-	-
27/2/2012 at 09:46:27	Axial loads	Structures	Mission	-	Requested	Not defined	Not defined	-	-
26/2/2011 at 19:49:17	Axial loads	Structures	Mission	-	External	Not defined	Not defined	-	-

Figure 60: Example of the application of INCREMENT by the system engineer to monitor the reason for a parameter blockage. In this case, the use of these visualization methods allows the detection of a correlation between the change in the parameter value of the Axial and Lateral loads and the "Problem" status

Step 6

Following the request of the system engineer, John, the "Mission Analysis" specialist, decides to develop a more accurate model to provide a better estimation of the Axial and Lateral loads depending on the type of launcher used in the mission. He looks at the manuals of different manufacturers and provides values of axial and lateral loads for four different launchers.

Since now there is a detailed model to define the values of these parameters, he sets the parameter dependency status to "External" (represented with the green color in the

dynamic DSM), the parameter assignment type to "Multiple" to show that there are several alternatives to the possible parameter values (represented with the extra text "multiple" next to the name of the parameter in the dynamic DSM), and the parameter assignment quality to "High quality" (represented with the horizontal pattern in the dynamic DSM) (Figure 61).

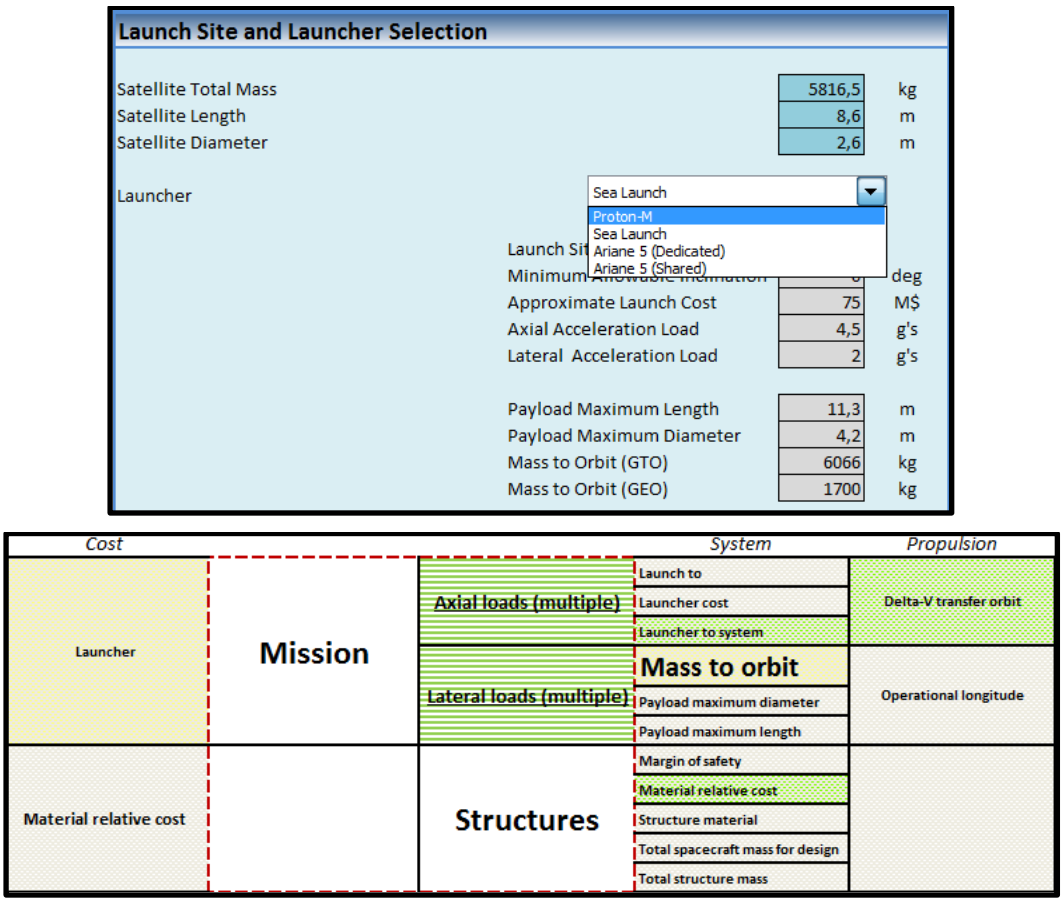


Figure 61: Example of the application of INCREMENT to develop a subsystem-level model to support a parameter value. Since the visual metaphor associated with the "Multiple" assignment type is the text "multiple", this is added after the name of the parameter. The quality of the assignment is changed to "High quality" and the parameter dependency status to "External", which is represented by an horizontal pattern and the green color, respectively

Step 7

Finally, Peter, the "Structures" domain specialist, reassesses the margin of safety in his subsystem-level analysis, and realizes that it is now positive for the "Proton-M" launcher. Since he agrees with the value of the parameter he sets the parameter completion status to "Complete", which is represented with the italic font on the dynamic DSM (Figure 62).

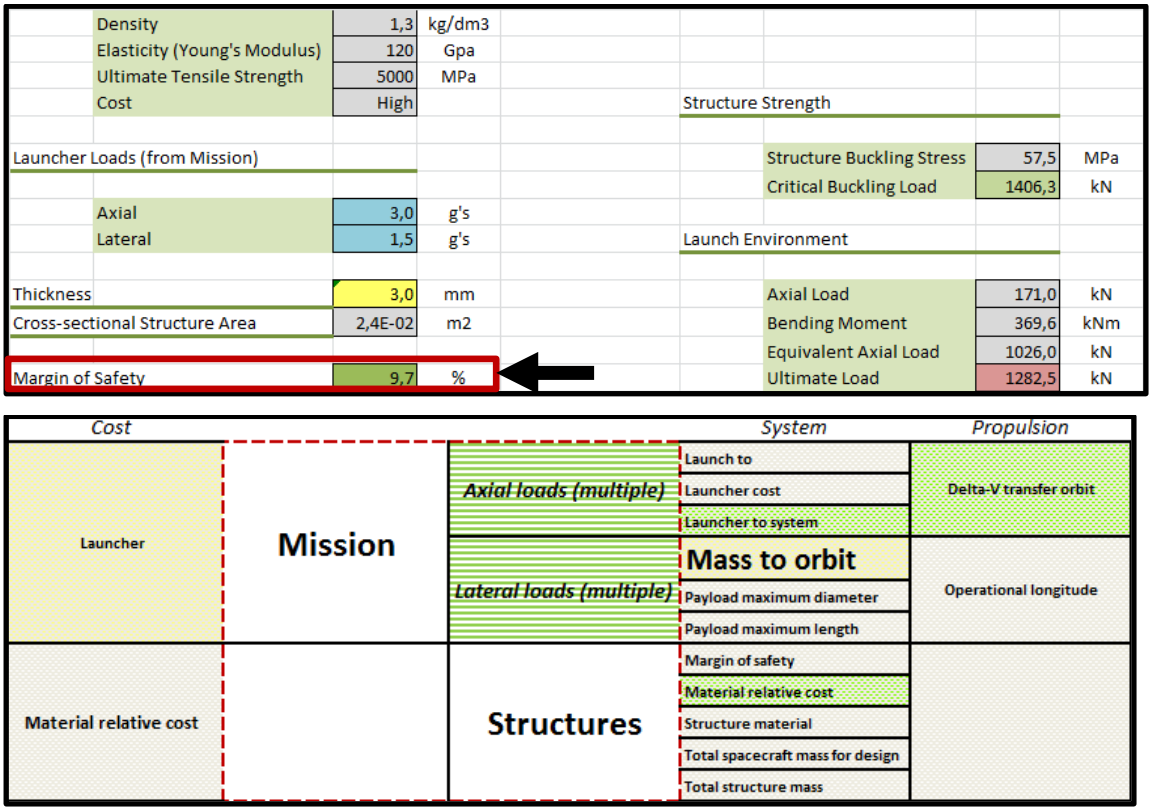


Figure 62: Example of the application of INCREMENT to assign the parameter completion status to "Complete" (represented with the italic font in the dynamic DSM)

6.6. Supporting the optimization stage

This section details the possible use of INCREMENT to support the generation and selection (narrowing down) of different design possibilities. These two capabilities are included in optimization stage as introduced in §4.4.

The design of a complex system is accomplished by a team of designers using mathematical models to determine the physical and functional characteristics of the system itself. A mathematical model is a set of relationships, i.e. equations, providing figures-of-merit on the performance(s) of the different subsystems when certain inputs are provided (Ridolfi et al. 2012).

Once the design parameters are defined and the subsystems level models linked, INCREMENT can use the assignments given to the input parameters of each subsystem to calculate the different design possibilities based on the models that were defined at the subsystem level. These results can be the departure point for a tradespace exploration strategy (G. Stump et al. 2009) (§4.4).

The example of the Earth-observation satellite is used one last time to demonstrate the possible use of INCREMENT to support the trade-space exploration and convergence of the design space.

In this example, three parameters are considered: "Number of Thrusters" (output of the Attitude Determination and Control System – ADCS), "Latitude" and "Longitude" (outputs of the Mission Analysis domain).

INCREMENT generates the dependency graph automatically based on the logic created on the calculation sheets where the relation between the number of thrusters and the other parameters is established, it determines the dependencies based on the formulas in the ADCS worksheet. The dependency graph shows that the "Number of Thrusters" depends on the values assigned to the "Latitude" and "Longitude" (Figure 63).

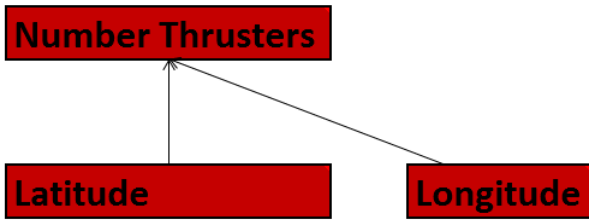
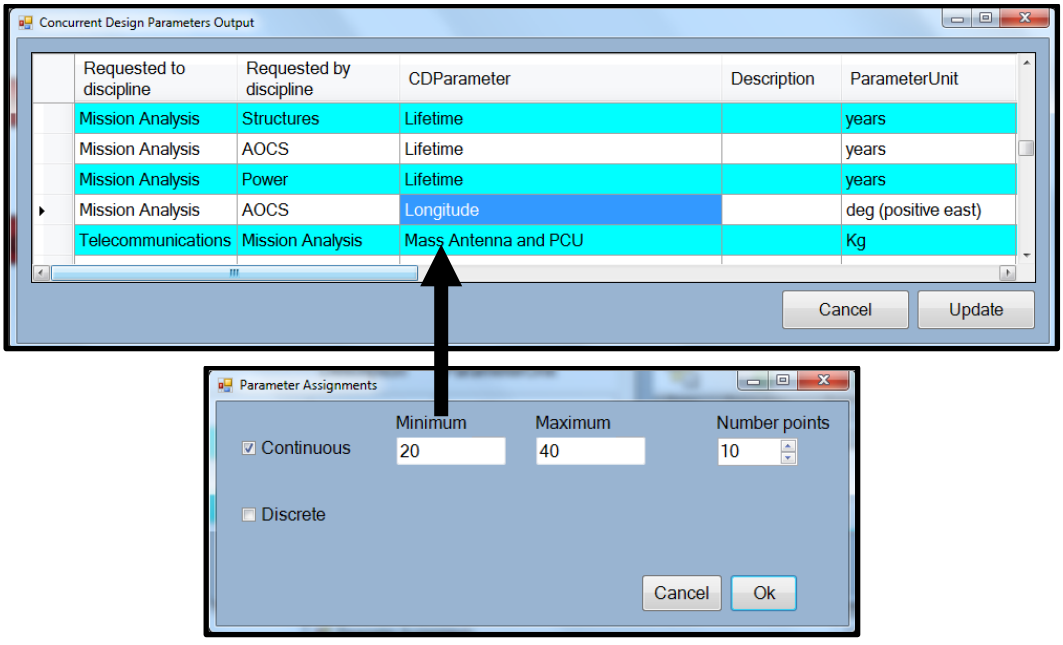


Figure 63: Example of the generation of a dependency graph for an Earth-observation satellite. In this case the "Number of Thrusters" definition depends on the value of "Latitude" and "Longitude"

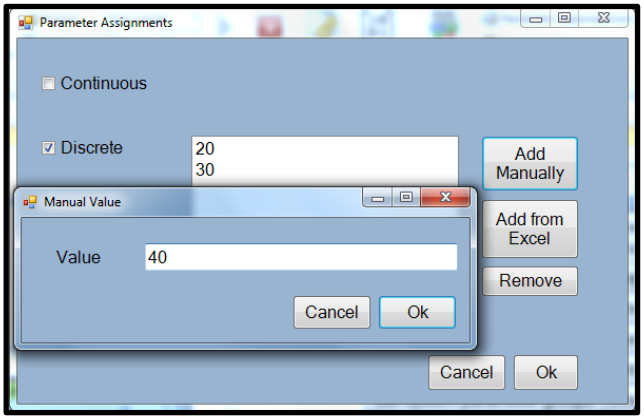
Initially, the person responsible for the Mission Analysis domain defines that the value of "Latitude" is fixed to a "single point" and that the "Longitude" can have a value between 20 and 40 deg and he/she decided to generate ten possibilities within this range. This results in ten design possibilities for the value of the number of thrusters (Figure 64).



Input :Latitude	Input :Longitude	Output: Number Thrusters
5	20	9
5	22,22222222	9
5	24,44444444	10
5	26,66666667	10
5	28,88888889	11
5	31,11111111	11
5	33,33333333	12
5	35,55555556	12
5	37,77777778	13
5	40	13

Figure 64: Example of the generation of design possibilities for the "Number of Thrusters" parameter of an Earth-observation satellite. In this case, since there are ten possibilities for the value of "Longitude", INCREMENT also generates ten possibilities for the "Number of Thrusters"

Afterwards the design process converges to only three discrete possibilities for the value of "Longitude" corresponding to 20, 30 and 40 deg. INCREMENT automatically updates the design space of the "Number of Thrusters" to only three possibilities (Figure 65).



Input :Latitude	Input :Longitude	Output: Number Thrusters
5	20	9
5	30	11
5	40	13

Figure 65: Example of the update of design possibilities for the "Number of Thrusters" parameter of an Earth-observation satellite. In this case, since the number of possible "Longitude" values is updated to three discrete options, INCREMENT also updates the number of possible values of the "Number of Thrusters" to three

6.7. Future enhancements

Despite the potential impact of the integrated design methodology, indicated by the capabilities of INCREMENT detailed in the earlier sections, without the case studies with experienced designers there will still be uncertainties related with its usability. Testing a software tool such as INCREMENT in «real-life» applications is particularly challenging due to the high number of resources involved. This requires a high level of software maturity. Since the assets involved (human and other) are usually quite expensive, it is

essential to achieve a stable software tool that can be used in complex applications, without compromising the quality of the process with bugs or poor interfaces.

Future work should focus on achieving an even more intuitive interface that, in addition to the functionalities detailed earlier, can also display an «acknowledging scheme» where the changes made by every subsystem only become definitive when acknowledged by the other related subsystems.

Another interesting issue that can help increase the dissemination of tools such as INCREMENT is the generation of automatic reports. Nowadays, conceptual design studies usually require the creation of burdensome asynchronous reports to justify the choices that led to the baseline solution. In this process a lot of intermediate information is lost which makes it tremendously inefficient for both the designer who is writing it, and future readers. INCREMENT already keeps track of all the design changes, along with their justifications. Generating automatic reports with this information can be very appealing to help reduce the loss of information and increase its acceptability from the designer's perspective.

With increased development resources, future work can focus on covering all the 4 stages of the integrated design methodology proposed in Chapter 4. The current functionalities of INCREMENT can support from the beginning of the general description stage to the end of the interaction stage. Additional functionalities can be developed to cover both the earlier general description and the optimization stages.

When looking at the general description stage, future work can focus on incorporating concept mapping technologies and/or integrating existing design modeling methods such as Harmony SE (§2.4.1) or OPM (§2.4.2) to help define the different system states that can then be integrated as design items in the current architecture of INCREMENT.

Finally, in order to expand the use of INCREMENT to the optimization stage, future research work can focus on enabling tradespace exploration strategies (§2.5.2.2) and/or MDO optimization methods (§2.5.2.3) using, as a departure point, the different design possibilities already obtained with the functionality described in §12.6.

[*This page intentionally left blank*]

7. CONCLUSIONS AND FUTURE WORK

This work focused on the field of collaborative design, an emerging discipline intrinsically linked with the new paradigm of «systems thinking», and the pervasive presence of computers. Yet, independently of the approaches or methods used to model systems, the human presence will continue to be critical in the inclusion of tacit knowledge and creativity into the design process by processing complex symbols and making decisions.

As the design process progresses and decisions accumulate, the ability to make changes is reduced and, simultaneously, the knowledge about the object of design increases. The main contributions of this work are related to the conceptual design phase, the most iterative part of the design process, where high level decisions that will impact later design phases are made. Enhancing the performance of this stage can dramatically impact the quality of the object of design and, help reduce the total expenditure, by improving information flow and avoiding rework activities.

The purpose of this chapter is to summarize and synthesize the results, contributions, and limitations of this work. The main contributions of this work to the growing body of work in the field of Engineering Systems are detailed in section 7.1, in terms of the literature reviewed, the results obtained, the methodologies proposed, and the tools developed. Then, section 7.2 answers the main research question of this work with the results obtained. In order to frame the applicability of this work, section 7.3 details the assumptions made and discusses its main limitations. Finally, section 7.4 offers several areas for future work to expand the impact of the research that this work has only just begun.

7.1. Contributions to the field of collaborative design

This work reviewed the main issues associated with the processes, products, and people behind the conceptual design of engineering systems exhibiting a significant level of complexity generated by an intricate blend of size, technical difficulty, and/or sociocultural matters.

On the process perspective, through a comprehensive exploration of the different design paradigms, design modeling methods, capabilities of current visualization methods, and

knowledge management strategies, the main weaknesses of current design processes were identified such as: lack of guidelines, lack of transparency, lack of traceability, low interoperability with the other design phases, or reduced perception of the current design status.

From the standpoint of the product, this research not only provided a system-level view of the design applications of current computer-based technologies, but also reviewed the drawbacks of commonly used collaborative design tools, such as: low efficiency communication strategies, low adaptability to the conceptual design where the level of required detail is lower, inability to deal with non-deterministic interactions, or lack of perception of the design process evolution.

With respect to the people issues, some issues were identified or confirmed by gathering the feedback from expert users of a concurrent engineering design environment through a survey designed to ascertain some of the main findings revealed in literature, and test potential different approaches to the problem. The survey showed a general dissatisfaction, 83% of the respondents revealed that current methodologies could or needed to be improved (these shortcomings were explored, through open questions, on a case by case analysis), it also identified the needs in terms of representation (hierarchical and design history views), the level of subsystem integration, and desired level of optimization.

By combining the issues detailed in literature with the results of the survey, this work elaborated a set of guidelines that were organized in the form of an integrated design methodology with the goal of enhancing the description, understanding, prediction, management, and/or change of complex engineering systems during the conceptual design phase.

This work proposed an integrated design methodology that: defined the common essential representation models for every type of complex engineering systems, established a shared integrated environment (encompassing the model definition and the design process management tasks), incorporated knowledge from earlier design processes, allowed transparent representation of the historic evolution of the design process, and tracked all design decisions.

This methodology relied on a simultaneous representation of the static structure, and the dynamic behavior of the system, without the complexity of a formal modeling language. It used new visualization methods to make the changes in the model and design evolution explicit, and facilitated the documentation of every decision (with a clear map between the

system representation and the actual design process). It was designed to not only aid system engineers, but also the different subsystem specialists involved in the design phase.

It provided clear guidelines in order to increase the performance of the conceptual design phase with four main stages: the system description stage, the requirements' mapping stage, the interaction stage, and the optimization stage. The methodology departed from a concept map of the system objective statement, followed by a definition of the different subsystems, and their associated requirements or constraints to feed a series of visualizations designed to monitor, and manage the different parameter requests. A variation of the Design Structure Matrix approach was presented to provide a global real-time perspective of the dependencies and the system coupling which, in addition to a static view of the structure of the system, allows the synchronous perception of the design process evolution, and its justification.

Some of the characteristics of this methodology were implemented by developing two tools: a predictive decision support tool using agent-based artificial neural network models to support early design decisions, and a higher level tool to support the overall design process designated as INtegrated Concurrent Real-time EnvironMENT (INCREMENT). The specific contributions of each of these tools are detailed in the following section by revisiting the initial research question.

7.2. Research question revisited

The main question on which this work was based was framed around its three components: the integrated design methodology, the predictive decision support tool, and the higher level tool to support the overall design process (INCREMENT).

The research question was: *In the context of complex engineering systems during the conceptual design phase, can the overall design solution and/or design process be enhanced by using the prediction capabilities of neural networks for decision support, and establishing a computer-based integrated design environment?*

The first part of the research question, related to the use of the prediction capabilities of neural networks during the conceptual design phase, was addressed in Chapter 5, which demonstrated a full analysis of the implementation of a predictive decision support tool. Based on this work, the answer to the first part of the research question is positive.

A prediction decision support tool was developed based on artificial neural network models and, by testing it on a total of six case studies performed with non-experienced

users, it was possible to measure a considerable gain in terms of the quality of the design solution when using the tool. The results obtained with 36 test users show a reduction in the error of the design decision, when using the proposed decision support tool. The gain in performance obtained with the tool, ranged from 13% for low complexity mathematical models, to 88% for more complex multivariate case studies.

The second part of the research question, regarding the establishment of an integrated design environment, was addressed in two different phases. In Chapter 4, this work proposed an integrated design methodology to overcome the main drawbacks of current design processes and modeling methodologies. Chapter 6 described the main characteristics of a support tool (INCREMENT) designed to implement the main ideas behind the integrated design methodology. Based on this work, even though the results were only qualitative, the answer to the second part of the research question can also be considered positive.

The proposed integrated design methodology considered the feedback from experienced users (Chapter 3) to propose a set of visualization methods and knowledge management strategies, organized in four different stages. The proposed methodology evolves from a simple objective and a set of requirements/constraints to a final design, through an evolutionary iterative process. It promotes a dynamic interactive evolution of objectives, requirements/constraints, subsystems, system states and design models.

The potential impact of the methodology was tested by including some of the main guidelines on INCREMENT, such as representation of the system as a whole, management of design documents integrated with the design process itself, or display of information about the role of the parameters of the system and their contribution to the design process.

These capabilities were demonstrated by simulating the design process of a telecommunications satellite system, which enabled a qualitative verification of the potential impact of some of the ideas behind the integrated design methodology. This is only a partial answer to this part of the research question because it was not possible to fully ascertain the quantitative performance gain of using INCREMENT in «real» collaborative design sessions, due to the limitations detailed in section 7.3.

7.3. Limitations

The limitations of this research are a result of its main assumptions. This work is specifically tailored for the conceptual design stage of complex engineering systems, it is important to acknowledge all the inherited implications of this fact.

The conceptual design phase is, by definition, wide and shallow, i.e. the analysis performed at this phase are usually very diverse, but with a low level of detail. It is the most influential stage of the design process, involving constant short-duration iterations to maximize design exploration and, finally, achieve a baseline solution consistent with all the requirements, constraints, and inherited subsystem level dependencies. This work considered that an integrated platform enables the design of all the subsystem models in the same platform and, at the same time, provides a system-level perspective of evolution of the design process, where every design decision can be tracked and justified. The methodology and tools proposed can be applied to downstream design applications but that would, in some cases, require the ability to include more detailed modeling methods, and/or software tools specific to more detail design analysis.

The complexity of the system is a main driver on the performance of the proposed contributions. The potential of the proposed methodologies and tools increases with the level of complexity, in terms of size (reflected in the number of items, subsystems, or dependencies), or sociocultural issues (reflected in inclusion of different organizations with different objectives, or people with different social and cultural backgrounds). If the system is relatively small, or can be designed by a small group of people with similar objectives and backgrounds, it might not be as advantageous to define a common representation, incorporate lessons learned from previous processes, or keep a historic evolution of the design process with full traceability of design decisions.

With respect to the predictive decision support tool introduced by this work, it is important to recall that the quality of the design suggestion is critically dependent on the quality of the historic dataset used. Even though the tool was designed to be as intuitive as possible, and independent of any previous background on neural networks, it is still highly reliable on the existence of a meaningful dataset, to which the tool can refer for training purposes. The quality of the suggestion is generally compromised when the range of the existing design solutions is very far from the desired range of predictions, making it only relevant for incremental decisions.

Finally, there is a limitation related with the quantitative performance gain of using a tool such as INCREMENT. In the present research, it was not possible to obtain a value on neither the design process performance, nor the quality of the design outcomes, because of the nature of conceptual design – highly dependent on the type of system and the background knowledge of the design team. This work focused on detailing the capabilities of the tool and their role within the overall integrated design methodology but, it was not possible to make any measurable case studies (such as the ones performed for the predictive decision support tool) representative of «real» collaborative design sessions. The potential impact was simply identified by a description of the different functionalities, by using an example of a telecommunications satellite system.

7.4. Future work

The findings described in this work provide the initial hints on how an integrated design methodology can increase the quality of the design process, in terms of final the outcome, and reduction of design duration (indirectly cost). These results only scratched the «tip of the iceberg» in the field of computer-enabled collaborative design applied to conceptual design studies, the number of potential improvements that can be made to the proposed methodology and support tools is, at least, as large as the number of possible applications.

As far as the methodology is concerned, some of the future developments can be: incorporating some of the already existing methods for requirements engineering (in the requirements' mapping stage), establishing guidelines to enable the inclusion of uncertainty into the design process (during the interaction stage), defining the typical interesting states to monitor during the interaction stage for different types of systems or, on the long term, proposing clearer instructions on choosing the level optimization for a particular system.

With respect to the predictive decision support tool, in order to overcome some of the drawbacks identified during the case studies (and the final survey performed to the subjects), the first enhancements should focus on accomplishing an even simpler tool. This could be achieved by abstracting the complexity of the setup process from the perspective of the designer. Future work can concentrate on including advices about possible reiterations regarding the filtering process and/or neural network definition (e.g. the tool may advice the user to filter the training data within a certain range depending on the prediction that he/she wishes to obtain).

Finally, regarding INCREMENT, future research can focus on establishing a case study with a significant level of complexity, but with well-defined and encapsulated tasks, that can be performed in a concurrent engineering environment with experienced users. This would provide a way to examine the quantitative performance gain, obtained by using the visualization methods, and knowledge management strategies proposed. In addition to this, there are a number of functionalities that can be developed, such as an «acknowledging scheme», where the changes made by every subsystem only become definitive when acknowledged by the other related subsystems, or the ability to generate automatic reports based on the decisions made throughout the design process.

[*This page intentionally left blank*]

8. REFERENCES

agi.com, Analytical Graphics, Inc. (AGI), analysis software for land, sea, air, and space. Available at: http://www.agi.com/ [Accessed August 31, 2011].

Agrawal, G. et al., 2004. Web-based visualization framework for decision-making in multidisciplinary design optimization. In *45th AIAA/ASME Conference*. pp. 19–22.

AHPproject, AHPproject - Free Web-Based Decision Support Tool. Available at: http://www.ahpproject.com/ [Accessed October 24, 2011].

Akay, M.F. et al., 2011. Artificial neural network-based model for predicting VO2max from a submaximal exercise test. *Expert Systems with Applications*, 38(3), pp.2007–2010.

Albert, S., 1998. Knowledge management: Living Up To the Hype? *Midrange Systems*, 11(13), p.52.

Anderson, E., 1935. The Irises of the Gaspe Peninsula. *Bulletin of the American Iris Society*, 59, pp.2–5.

Arbib, M., 1995. *The handbook of brain theory and neural networks*, Cambridge Mass.: MIT Press.

Argote, L., 2000. Knowledge Transfer: A Basis for Competitive Advantage in Firms. *Organizational Behavior and Human Decision Processes*, 82(1), pp.150–169.

Asaduzzaman, M., Shahjahan, M. & Murase, K., 2009. Faster training using fusion of activation functions for feed forward neural networks. *International Journal of Neural Systems*, 19(06), p.437.

astos.de, Astos Solutions. Available at: http://www.astos.de/ [Accessed August 31, 2011].

Austin, S.A. et al., 2000. Application of the analytical design planning technique to construction project management.

AutoVue, 2011. Oracle's AutoVue Enterprise Visualization Solutions | Oracle Products. Available at: http://www.oracle.com/us/products/applications/autoVue/index.html [Accessed August 12, 2011].

Avnet, M.S. & Weigel, A.L., 2010. An application of the Design Structure Matrix to Integrated Concurrent Engineering. *Acta Astronautica*, 66(5-6), pp.937–949.

Axelrod, R., 1997. *The complexity of cooperation : agent-based models of competition and collaboration*, Princeton N.J.: Princeton University Press.

Balamuralikrishna, R. et al., 2000. The relevance of concurrent engineering in industrial technology programs. *Journal of Industrial Technology*, 16(3), pp.1–5.

Bandecchi, M., Melton, B. & Gardini, B., 2000. The ESA/ESTEC Concurrent Design Facility. In *Systems engineering - a key to competitive advantage for all industries*. European Systems Engineering Conference (EuSEC 2000). München: Utz.

Bederson, B., 2003. *The craft of information visualization : readings and reflections*, Amsterdam: Morgan Kaufmann.

Bidarra, R., van den Berg, E. & Bronsvoort, W.F., 2002. A collaborative feature modeling system. *Journal of Computing and Information Science in Engineering*, 2(3), p.192.

Blanchard, B., 2008. *System engineering management, fourth edition*, Hoboken, N.J.: John Wiley & Sons.

Boccaletti, S. et al., 2006. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5), pp.175–308.

Bontis, N., Serenko, A. & Biktimirov, E., 2006. MBA knowledge management course: is there an impact after graduation? *International Journal of Knowledge and Learning*, 2(3/4), p.216.

Box, G., 1987. *Empirical model-building and response surfaces*, New York: Wiley.

Bresciani, S. & Eppler, M.J., 2009a. The benefits of synchronous collaborative information visualization: Evidence from an experimental evaluation. *Visualization and Computer Graphics, IEEE Transactions on*, 15(6), pp.1073–1080.

Bresciani, S. & Eppler, M.J., 2009b. The Risks of Visualization. *Institute for corporate communication*, University of Lugano(ICA Working Paper).

Breslow, N., 1970. A generalized Kruskal-Wallis test for comparing K samples subject to unequal patterns of censorship. *Biometrika*, 57(3), p.579.

Brooks, F., 2010. *The design of design : essays from a computer scientist*, Upper Saddle River NJ: Addison-Wesley.

Brown, D.C. et al., 1995. SINE: support for single function agents. *Proceedings of AIENG*, 95.

Browning, T.R., 2001. Applying the design structure matrix to system decomposition and integration problems: a review and new directions. *Engineering Management, IEEE Transactions on*, 48(3), pp.292–306.

cadence.com, Cadence PSpice A/D and Advanced Analysis. Available at: http://www.cadence.com/products/orcad/pspice_simulation [Accessed August 31, 2011].

Cardei, I., Fonoage, M. & Shankar, R., 2008. Model Based Requirements Specification and Validation for Component Architectures. In *The 2nd IEEE International Systems Conference*.

CCTA, 2000. *SSADM foundation*, London: Stationery Office.

Chapman, W., 1992. *Engineering modeling and design*, Boca Raton Fla.: CRC Press.

Chen, P.P., 1976. The entity-relationship model—toward a unified view of data. *ACM Transactions on database systems*, 1(1), pp.9–36.

Chen, Y.J., Chen, Y.M. & Chu, H.C., 2008. Enabling collaborative product design through distributed engineering knowledge management. *Computers in Industry*, 59(4), pp.395–409.

Cmap, CmapTools - Home Page Cmap.html. Available at: http://cmap.ihmc.us/ [Accessed October 19, 2011].

ConceptStation, 2011. ConceptStation (RealityWave Inc) - Development/Rich Media - Datamation Product Watch. Available at: http://products.datamation.com/development/rich_media/982097698.html [Accessed August 12, 2011].

Cross, N., 1992. *Research in design thinking* Technische Hogeschool Delft., Delft: Delft University Press.

Cutkosky, M.R. et al., 1993. PACT: An experiment in integrating concurrent engineering systems. *Computer*, 26(1), pp.28–37.

dakota.sandia.gov, The DAKOTA Project - Software Design. Available at: http://dakota.sandia.gov/ [Accessed August 29, 2012].

Danilovic, M. & Browning, T.R., 2007. Managing complex product development projects with design structure matrices and domain mapping matrices. *International Journal of Project Management*, 25(3), pp.300–314.

Date, C., 2004. *An introduction to database systems* 8th ed., Boston: Pearson/Addison Wesley.

Desouza, K., 2005. *New frontiers of knowledge management*, Houndmills Basingstoke; New York: Palgrave Macmillan.

Dhaliwal, J.S. & Benbasat, I., 1996. The Use and Effects of Knowledge-Based System Explanations: Theoretical Foundations and a Framework for Empirical Evaluation. *Information Systems Research*, 7(3), pp.342–362.

Diller, N.P., 2002. *Utilizing multiple attribute tradespace exploration with concurrent design for creating aerospace systems requirements*. Massachusetts Institute of Technology.

Dori, D., 2002. *Object-process methodology : a holistics systems paradigm*, Berlin: Springer.

Dorst, K., 1995. Comparing paradigms for describing design activity. *Design Studies*, 16(2), pp.261–274.

Dowlatshahi, S., 1994. A comparison of approaches to concurrent engineering. *The International Journal of Advanced Manufacturing Technology*, 9(2), pp.106–113.

Dowlatshahi, S., 1999. A modeling approach to logistics in concurrent engineering. *European Journal of Operational Research*, 115(1), pp.59–76.

Du, X. & Chen, W., 2000. Towards a Better Understanding of Modeling Feasibility Robustness in Engineering Design. *Journal of Mechanical Design*, 122(4), p.385.

eclipse.org, Eclipse - The Eclipse Foundation open source community website. Available at: http://www.eclipse.org/ [Accessed August 22, 2011].

ecosimpro.com, EcosimPro: Continuous and Discrete Modelling and Simulation Software. Available at: http://www.ecosimpro.com/ [Accessed August 31, 2011].

ECSS-TM-E-10-25A, 2010. Space engineering: Engineering design model data exchange (CDF).

Eddy, W.F. & Mockus, A., 1995. Dynamic visualization in modeling and optimization of ill-defined problems: case studies and generalizations. *State of the Art in Global Optimization: Computational Methods and Applications*, 7, pp.499–520.

eDrawings, 2011. Free eDrawings Viewer for SolidWorks, DWG and DXF files. Available at: http://www.edrawingsviewer.com/ [Accessed August 12, 2011].

Edwards, K., 2001. Epistemic communities, situated learning and open source software development. *Department of Manufacturing Engineering and Management*, Technical University of Denmark.

Emissions, E.C., 2009. *Reducing CO2 emissions from passenger cars - Documentation - Climate Action - European Commission*, Available at: http://ec.europa.eu/clima/documentation/transport/vehicles/cars_en.htm.

English, Kenneth W. & Bloebaum, C.L., 2008. Visual Dependency Structure Matrix for Multidisciplinary Design Optimization Tradeoff Studies. *Journal of Aerospace Computing, Information, and Communication*, 5, pp.1–21.

Enke, D. & Thawornwong, S., 2005. The use of data mining and neural networks for forecasting stock market returns. *Expert Systems with Applications*, 29(4), pp.927–940.

ENOVIA, 2011. ENOVIA - PLM Solutions - Portfolio - Dassault Systèmes. Available at: http://www.3ds.com/products/enovia/products/ [Accessed August 12, 2011].

Eppinger, S., 2001. Innovation at the speed of information. *Harvard Business Review*.

Eppinger, S.D. et al., 1994. A model-based method for organizing tasks in product development. *Research in Engineering Design*, 6(1), pp.1–13.

Ertas, A., 1996. *The engineering design process* 2nd ed., New York: John Wiley & Sons.

ESA - CDF, 2011. ESA - CDF. Available at: http://www.esa.int/esaMI/CDF/ [Accessed August 17, 2011].

Estefan, J.A., 2007. Survey of model-based systems engineering (MBSE) methodologies. *Incose MBSE Focus Group*, 25.

Findlay, R. et al., 2011. Implementation of concurrent engineering to Phase B space system design. *CEAS Space Journal*. Available at: http://www.springerlink.com/index/10.1007/s12567-011-0013-y [Accessed August 23, 2011].

Forman., E.H. & Gass, S.I., 2001. The Analytic Hierarchy Process--An Exposition. *Operations Research*, 49(4), pp.469–486.

Fortescue, P., 2003. *Spacecraft systems engineering* 3rd ed., New York: J. Wiley.

French, M., 1985. *Conceptual design for engineers* 2nd ed., London: Design Council.

Geman, S., Bienenstock, E. & Doursat, R., 1992. Neural Networks and the Bias/Variance Dilemma. *Neural Computation*, 4(1), pp.1–58.

Gil, P.J.S., Rosa, P.M.B. & Ferreira, I.M.L., 2010. Modern approaches in the design of complex aerospace systems. *Journal of Aerospace Engineering, Sciences and Applications*, 2(1), pp.15–26.

Gill, J., 1977. Computational Complexity of Probabilistic Turing Machines. *SIAM Journal on Computing*, 6(4), p.675.

Gloor, P., 2006. *Swarm creativity : competitive advantage through collaborative innovation networks*, Oxford: Oxford University Press.

Goldfinger, A., 2000. A knowledge-based approach to spacecraft distributed modeling and simulation. *Advances in Engineering Software*, 31(8-9), pp.669–677.

Hadland, T., 2000. *The Moulton bicycle : (the story from 1957-1981)* 2nd ed., Coventry: T. Hadland.

Hammond, J.M. et al., 2005. Distributed collaborative design teams: media effects on design processes. *International Journal of Human Computer Interaction*, 18(1), pp.145–166.

Hartley, J., 1992. *Concurrent engineering : shortening lead times, raising quality, lowering costs*, Portland: Poductivity.

Heaton, J., 2008. *Introduction to Neural Networks for Java, 2nd Edition*, Heaton Research, Inc.

Heer, J., Card, S.K. & Landay, J.A., 2005. Prefuse: a toolkit for interactive information visualization. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. pp. 421–430.

Van Heijst, G., van der Spek, R. & Kruizinga, E., 1997. Corporate memories as a tool for knowledge management. *Expert Systems with Applications*, 13(1), pp.41–54.

Hicks, B.J. et al., 2002. A framework for the requirements of capturing, storing and reusing information and knowledge in engineering design. *International journal of information management*, 22(4), pp.263–280.

Hopfield, J.J. & Tank, D.W., 1985. Neural Computation of Decision in Optimization Problems. *Biological Cybernetics*, 52, pp.141–142.

Huthwaite, B., 1988. Designing in quality. *Journal of Quality Technology*, 11(27), pp.34–35.

IBM DOORS, IBM - Rational DOORS - Software. Available at: http://www-01.ibm.com/software/awdtools/doors/ [Accessed October 18, 2011].

ibm.com, IBM - Rational Tau - Software. Available at: http://www-01.ibm.com/software/awdtools/tau/ [Accessed August 25, 2011].

incose.org, International Council on Systems Engineering Website. Available at: http://www.incose.org/ [Accessed August 23, 2011].

iosotech.com, "Sigma Technology". Novel Optimization Strategy - IOSO. Available at: http://iosotech.com/ [Accessed August 29, 2011].

Jameson, A., 1999. Re-Engineering the Design Process Through Computation. *Journal of Aircraft*, 36(1), pp.36–50.

Jones, C.V., 1994. Visualization and Optimization. *INFORMS Journal on Computing*, 6(3), pp.221–257.

Kahaner, D. & Lu, S., 1993. First CIRP International Workshop on Concurrent Engineering for Product Realization. *Concurrent Engineering Research in Review*, (5), pp.6–14.

Keeney, R., 1976. *Decisions with multiple objectives : preferences and value tradeoffs*, New York: Wiley.

Kemp, C. & Tenenbaum, J.B., 2008. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31), p.10687.

King, N. & Majchrzak, A., 1996. Concurrent engineering tools: are the human issues being ignored? *Engineering Management, IEEE Transactions on*, 43(2), pp.189–201.

Kisi, O., 2004. Multi-layer perceptrons with Levenberg-Marquardt training algorithm for suspended sediment concentration prediction and estimation. *Hydrological Sciences Journal*, 49(6), pp.1025–1040.

Kossiakoff, A., 2003. *Systems engineering : principles and practices*, New York N.Y.: Wiley.

Kurzweil, R., 2005. *The singularity is near : when humans transcend biology*, London: Gerald Duckworth.

Kusiak, A., 1993. *Concurrent engineering : automation, tools, and techniques*, New York: Wiley.

Lai, H.-H., Lin, Y.-C. & Yeh, C.-H., 2005. Form design of product image using grey relational analysis and neural network models. *Computers & Operations Research*, 32, pp.2689–2711.

Lanza, M., 2003. CodeCrawler - A lightweight software visualization tool. *Proceedings of VISSOFT 2003 (2nd International Workshop on Visualizing Software for Understanding and Analysis)*, pp.51–52.

Lapicque, L., 1907. Quantitative investigations of electrical nerve excitation treated as polarization. 1907. *Biological cybernetics*, 97(5-6), pp.341–9.

Larman, C. & Basili, V.R., 2003. Iterative and incremental development: a brief history. *Computer*, 36(6), pp.47–56.

Larson, W., 1997. *Space mission analysis and design* 2. ed., Dordrecht: Kluwer Academic Publishers.

Laudon, K., 2011. *Essentials of management information systems* 9th ed., Upper Saddle River NJ: Prentice Hall.

Li, W., 2006. *Integrated and collaborative product development environment : technologies and implementations*, Singapore: World Scientific.

Li, W.D. & Qiu, Z.M., 2006. State-of-the-art technologies and methodologies for collaborative product development systems. *International Journal of Production Research*, 44(13), pp.2525–2559.

Liao, S., 2003. Knowledge management technologies and applications–literature review from 1995 to 2002. *Expert systems with applications*, 25(2), pp.155–164.

Liebowitz, J. & Wright, K., 1999. Does measuring knowledge make "cents"? *Expert systems with applications*, 17(2), pp.99–103.

Lim, T.S., Loh, W.Y. & Shih, Y.S., 2000. A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine learning*, 40(3), pp.203–228.

Lindemann, U., 2009. *Structural complexity management an approach for the field of product design*, Berlin: Springer.

Lindemann, U., Maurer, M. & Braun, T., 2009. Complexity in the context of product design. In *Structural Complexity Management*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 21–42.

linux.com, Linux.com | The source for Linux information. Available at: http://www.linux.com/ [Accessed August 22, 2011].

Loureiro, G. & Leaney, P.G., 2003. A systems and concurrent engineering framework for the integrated development of space products. *Acta Astronautica*, 53(12), pp.945–961.

Ma, Y.-S., Chen, G. & Thimm, G., 2008. Paradigm shift: unified and associative feature-based concurrent and collaborative engineering. *Journal of Intelligent Manufacturing*, 19(6), pp.625–641.

Magee, C. & de Weck, O., 2004. Complex system classification. In *Fourteenth Annual International Symposium of the International Council On Systems Engineering (INCOSE)*.

MagicDraw, UML | UML Modeling Tool | UML Designing Tool | Graphical Modelling Tool from No Magic. Available at: https://www.magicdraw.com/ [Accessed October 19, 2011].

Malhotra, Y., 2001. *Knowledge management and business model innovation*, Hershey [PA]: Idea Group Pub.

Marquardt, D.W., 1963. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *SIAM Journal on Applied Mathematics*, 11(2), p.431.

mathworks.com, MATLAB - The Language Of Technical Computing. Available at: http://www.mathworks.com/products/matlab/index.html [Accessed August 31, 2011].

McConnell, S., 1996. *Rapid development : taming wild software schedules*, Redmond Wash.: Microsoft Press.

McCord, K.R., Eppinger, S.D. & Management, S.S. of, 1993. *Managing the integration problem in concurrent engineering*, Alfred P. Sloan School of Management, Massachusetts Institute of Technology.

Messac, A. & Chen, X., 2000. Visualizing the optimization process in real-time using physical programming. *Engineering Optimization*, 32(6), pp.721–747.

Mistree, F., Smith, W. & Bras, B., 1993. A decision-based approach to concurrent design. *Concurrent Engineering: Contemporary Issues and Modern Design Tools. HR Parsaei & WG Sullivan (eds.). Chapman & Hall, London*, pp.127–158.

modefrontier.com, modeFRONTIER - Home. Available at: http://www.modefrontier.com/homeMF.html [Accessed August 29, 2011].

Molina, A. et al., 1995. A review of computer-aided simultaneous engineering systems. *Research in Engineering Design*, 7(1), pp.38–63.

Morse, E. et al., 2006. Next-generation concurrent engineering: Developing models to complement point designs. In *Aerospace Conference, 2006 IEEE*. p. 15.

NASA, 2007. NASA Systems Engineering Handbook.

Nelson, M., 1994. *A practical guide to neural nets*, Reading Mass.: Addison-Wesley.

Nevins, J., 1989. *Concurrent design of products and processes : a strategy for the next generation in manufacturing*, New York: McGraw-Hill.

Newell, A., 1972. *Human problem solving*, Englewood Cliffs N.J.: Prentice-Hall.

North, C., 2006. Toward measuring visualization insight. *IEEE Computer Graphics and Applications*, pp.6–9.

Omar, H.M., Bakar, N.A.A.A. & Nor, N.M., 2009. Virtual Handycraft: Interactivity in 3D Product Using VRML. In *2009 Second International Conference on Computer and Electrical Engineering*. 2009 Second International Conference on Computer and Electrical Engineering. Dubai, UAE, pp. 634–638.

OMG, OMG Document -- ad/97-08-11 (Set of PDF versions of ad/97-08-02 through ad/97-08-09, v1.1). Available at: http://www.omg.org/cgi-bin/doc?ad/97-08-11 [Accessed August 25, 2011].

opcat.com, Opcat Systems. Available at: http://www.opcat.com/ [Accessed August 25, 2011].

Paliwal, M. & Kumar, U.A., 2009. Neural networks and statistical techniques: A review of applications. *Expert Systems with Applications*, 36(1), pp.2–17.

Panchal, J.H. et al., 2007. Collaborative applications, product design and manufacturing methodologies and applications - Leveraging design process related intellectual capital – A key to enhancing enterprise agility. In *Springer Series in Advanced Manufacturing*. pp. 202–233.

Papyrus, Papyrus. Available at: http://www.eclipse.org/modeling/mdt/papyrus/ [Accessed October 19, 2011].

Parnas, D., 2003. A rational design process: how and why to fake it. *IEEE Computer*.

Parsons, F.. & Wirsching, P.., 1982. A Kolmogorov - Smirnov goodness-of-fit test for the two-parameter weibull distribution when the parameters are estimated from the data. *Microelectronics Reliability*, 22(2), pp.163–167.

Paulk, M., 1995. *The capability maturity model : guidelines for improving the software process*, Reading Mass.: Addison-Wesley Pub. Co.

Pedregal, P., 2003. *Introduction to optimization*, New York: Springer.

powersimtech.com, PSIM Software by Powersim Inc. Available at: http://www.powersimtech.com/ [Accessed August 31, 2011].

ptc.com, PTC – Mathcad – Engineering Calculations Software. Available at: http://www.ptc.com/products/mathcad/ [Accessed August 31, 2011].

ptc.com, 2011. PTC – Windchill – Business Process Management (BPM) – Collaboration Software. Available at: http://www.ptc.com/products/windchill/ [Accessed August 12, 2011].

Ralph, P., 2010. Comparing two software design process theories. *Global Perspectives on Design Science Research*, pp.139–153.

reactiondesign.com, Reaction Design - CHEMKIN. Available at: http://www.reactiondesign.com/products/open/chemkin.html [Accessed August 31, 2011].

redcedartech.com, The Next Generation in Design Optimization | Red Cedar Technology. Available at: http://www.redcedartech.com/ [Accessed August 29, 2011].

Reddy, Y.V.R. et al., 1993. Computer Support for Concurrent Engineering. *IEEE Computer*, 26(1), pp.12–16.

Reich, Y. et al., 1999. Building agility for developing agile design information systems. *Research in Engineering Design*, 11(2), pp.67–83.

Rhyne, T.-M., 2009. Toward Measuring Visualization Insight. *IEEE Computer Graphics and Applications*, (May/June 2006).

Richardson, R. & Dunne, C., 2010. OCDS – A Cross-Industry Concurrent Design Platform. In SECESA 2010 - 4th International Workshop on System & Concurrent Engineering for Space Applications. Lausanne.

Ridolfi, G., Mooij, E. & Corpino, S., 2012. Complex-Systems Design Methodology for Systems-Engineering Collaborative Environment. In B. Cogan, ed. *Systems Engineering - Practice and Theory*. InTech. Available at: http://www.intechopen.com/books/systems-engineering-practice-and-theory/complex-systems-design-methodology-for-systems-engineering-collaborative-environment [Accessed June 26, 2012].

Robbins, H., 1952. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58(5), pp.527–536.

Rodgers, J., 1999. DeMAID: A Software Tool for Analyzing and Optimizing the DSM.

Rodgers, J.L., Salars, A.O. & Weston, R.P., 1999. A web-based monitoring system for multidisciplinary design projects. *American Institute of Aeronautics and Astronautics*.

Rodriguez, K. & Al-Ashaab, A., 2005. Knowledge web-based system architecture for collaborative product development. *Computers in Industry*, 56(1), pp.125–140.

Ross, A.M. et al., 2004. Multi-attribute tradespace exploration as front end for effective space system design. *Journal of Spacecraft and Rockets*, 41(1), pp.20–28.

Roth, R., Field, F. & Clark, J., 1994. Materials selection and multi-attribute utility analysis. *Journal of Computer-Aided Materials Design*, 1(3), pp.325–342.

Rowley, J., 2007. The wisdom hierarchy: representations of the DIKW hierarchy. *Journal of Information Science*, 33(2), pp.163–180.

Saaty, T., 2008. *Group decision making : drawing out and reconciling differences*, Pittsburgh PA: RWS Publications.

Saaty, T., 2009. *Theory and applications of the analytic network process : decision making with benefits, opportunities, costs, and risks* 2nd printing with corr., Pittsburgh Penn.: RWS Publications.

Saltelli, A. & Annoni, P., 2010. How to avoid a perfunctory sensitivity analysis. *Environmental Modelling & Software*, 25(12), pp.1508–1517.

dos Santos, A.C. et al., 1997. SACE-CSCW: a synchronous asynchronous common environment for computer supported cooperative work to aid concurrent engineering processes. In *Computer Science Society, 1997. Proceedings., XVII International Conference of the Chilean*. pp. 218–226.

Sata, T. et al., 1985. Designing Machine Assembly Structure Using Geometric Constraints in Product Modelling. *CIRP Annals - Manufacturing Technology*, 34(1), pp.169–172.

Schon, D., 2000. *Reflective practioner : How professionals think in action.*, New York: BasicBooks.

Seo, K.K. et al., 2002. Approximate estimation of the product life cycle cost using artificial neural networks in conceptual design. *The international journal of advanced manufacturing technology*, 19(6), pp.461–471.

Shen, W., 2001. *Multi-agent systems for concurrent intelligent design and manufacturing*, London; New York: Taylor & Francis.

Shen, W. & Barther, J.-P., 1995. DIDE: A Multi-Agent Environment for Engineering Design. In *Proceedings of the First International Conference on Multiagent Systems*.

Shen, W., Hao, Q. & Li, W., 2008. Computer supported collaborative design: Retrospective and perspective. *Computers in Industry*, 59(9), pp.855–862.

Simpson, T. et al., 2007. Impact of response delay and training on user performance with text-based and graphical user interfaces for engineering design. *Research in Engineering Design*, 18(2), pp.49–65.

SmartDraw, Concept Map - Visualize your ideas with mind maps. Available at: http://www.smartdraw.com/specials/ppc/ [Accessed October 19, 2011].

smashingmagazine.com, Data Visualization: Modern Approaches - Smashing Magazine. Available at: http://www.smashingmagazine.com/2007/08/02/data-visualization-modern-approaches/ [Accessed August 19, 2011].

Sommerville, I., 1997. *Requirements engineering : a good practice guide*, Chichester: Wiley.

Spinfire, 2011. Spinfire - CAD Solution System - Actify. Available at: http://www.actify.com/products/spinfire-product-suite-cad-solution-system [Accessed August 12, 2011].

Sprow, E., 1992. Chrysler's concurrent engineering challenge. *Manufacturing Engineering*, 108(4), pp.35–42.

Stark, J., 2011. *Product lifecycle management: 21st century paradign for product realisation (decision engineering)* 2nd ed., Available at: Springer.

Starnone, M.D., 2005. Space Systems Engineering Technology Improvements using STARMAD (Space Tool for Advanced & Rapid Mission Analysis & Design) in the Design Process of a Space Mission.

Steeb, W., 2008. *The nonlinear workbook : chaos, fractals, cellular automata, neural networks, genetic algorithms, gene expression programming, support vector machine, wavelets, hidden Markov models, fuzzy logic with* 4th ed., Singapore; Hackensack NJ: World Scientific.

steptools.com, STEP Tools - Products. Available at: http://www.steptools.com/products/index.html [Accessed August 23, 2011].

Streamline, 2011. Autodesk - Autodesk Streamline. Available at: http://usa.autodesk.com/adsk/servlet/pc/ [Accessed August 12, 2011].

Stump, G. et al., 2009. Visual Steering Commands for Trade Space Exploration: User-Guided Sampling With Example. *Journal of Computing and Information Science in Engineering*, 9(4), p.044501.

Sutton, G., 2010. *Rocket propulsion elements* 8th ed., Hoboken N.J.: Wiley.

Teamcenter, 2011. Teamcenter Engineering Process Management: Siemens PLM Software. Available at: http://www.plm.automation.siemens.com/ [Accessed August 12, 2011].

Thomas, J. & Cook, K., 2005. *Illuminating the Path: The R&D Agenda for Visual Analytics*,

Thouvenin, I. et al., 2005. Knowledge integration in early design stages for collaboration on a virtual mock up (PDF). In *Proceedings of the Ninth International Conference on Computer Supported Cooperative Work in Design*.

Tomiyama, T., 2006. Collaborative product development in ill-structured problem domains. In *Proceedings of the 10th International Conference on Computer Supported Cooperative Work in Design*.

Tufte, E., 2006. *Beautiful evidence*, Cheshire Conn.: Graphics Press.

Turino, J., 1992. *Managing concurrent engineering : buying time to market : a definitive guide to improved competitiveness in electronics design and manufacturing*, New York: Van Nostrand Reinhold.

Ulrich, K. & Pearson, S., 1993. Does Product Design Really Determine. 80% of Manufacturing Cost? *Working paper. Alfred P. Sloan School of Management*.

Vellido, A., Lisboa, P.J.. & Vaughan, J., 1999. Neural networks in business: a survey of applications (1992-1998). *Expert Systems with Applications*, 17(1), p.51.

Vira, C., 1983. *Multiobjective decision making : theory and methodology*, New York: North Holland.

visualcomplexity.com, visualcomplexity.com | A visual exploration on mapping complex networks. Available at: http://www.visualcomplexity.com/vc/ [Accessed August 21, 2011].

visual-literacy.org, A periodic table of visualization methods. Available at: http://www.visual-literacy.org/periodic_table/periodic_table.html [Accessed August 19, 2011].

w3.org, Extensible Markup Language (XML). Available at: http://www.w3.org/XML/ [Accessed August 23, 2011].

Wainer, H., 2004. How to display data badly. , 2(38), pp.137–147.

Weber, C., 2005. What is "complexity"? In *Proceedings of the 15th International Conference on Engineering Design (ICED 05)*. Melbourne.

Weilkiens, T., 2007. *Systems engineering with SysML UML modeling, analysis, design*, Burlington MA: Morgan Kaufmann.

Whitney, D. et al., 1999. *Introducing knowledge-based engineering into an interconnected product development process*,

Wiig, K., 1994. *Knowledge management : the central management focus for intelligent-acting organizations*, Arlington Tex.: Schema Press.

Wiig, K.M., 1997. Knowledge management: where did it come from and where will it go? *Expert systems with applications*, 13(1), pp.1–14.

Womack, J., 1991. *The machine that changed the world : how Japan's secret weapon in the global auto wars will revolutionize western industry* 1st HarperPerennial ed., New York NY: HarperPerennial.

Yassine, A.A., 2006. An introduction to modeling and analyzing complex product development processes using the design structure matrix (DSM) method. *Urbana*, 51, p.61801.

Yuan Miao & Zhi-Qiang Liu, 2000. On causal inference in fuzzy cognitive maps. *IEEE Transactions on Fuzzy Systems*, 8(1), pp.107–119.

Yukish, M., Stump, G.M. & Lego, S., 2007. Visual steering and trade space exploration. In *Aerospace Conference, 2007 IEEE*. pp. 1–9.

zemax.com, Home - ZEMAX: Software for Optical System Design. Available at: http://www.zemax.com/ [Accessed August 31, 2011].

Zins, C., 2007. Conceptual approaches for defining data, information, and knowledge. *Journal of the American Society for Information Science and Technology*, 58(4), pp.479–493.

Zopounidis, C., 2010. *Handbook of Multicriteria Analysis*, Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg.

9. APPENDIX 1

Survey space systems integrated design and analysis

(the survey was available online through a web-based survey engine)

Thank you very much for your attention. The following survey has 26 questions and shall not require more than 10/15 minutes of your time.

This survey is entirely voluntary. You may answer as few or as many questions as you would like. Care will be taken to ensure confidentiality of the responses.

In this section, please let us know some basic information about yourself as well as your background

General Questions

1. Age:

□ < 20 □ 20-30 □ 30-40 □40-50 □ 50-60 □60-70 □70-80 □ > 80

2. Gender:

□ Male □ Female

3. Educational Background

□ High School
□ Bachelors (BSc)
□ Masters (MSC or MEng)
□ PhD
□ Other, please specify

4. Please choose the area which better relates to your work (several answers can be chosen):

□ Instruments
□ Space Systems
□ Science
□ Propulsion
□ Materials
□ Mechanisms
□ Structures
□ Thermal
□ GNC
□ Mission Analysis

□ Ground Segment and Operations
□ Communications
□ Simulation
□ Human Space Flight
□ Programmatics
□ Risk Analysis
□ Cost Analysis
□ Power
□ Other, please specify

5. Can you briefly describe your work? (Open question)

The need of an integrating tool

A system can be broadly called complex in nature when it cannot be reduced to a sum of simpler parts or when it is difficult to do that analysis. Any sufficiently complicated system, with many elements and/or interactions, can be considered complex as its analysis will always be hard to accomplish.

6. In your opinion, which of the following systems are complex (from 1 to 5, 5 being the most complex one)?

	1	2	3	4	5
A rocket nozzle	□	□	□	□	□
An antenna	□	□	□	□	□
A small launcher	□	□	□	□	□
A telecommunications satellite	□	□	□	□	□
A satellite navigation system	□	□	□	□	□
A manned spacecraft	□	□	□	□	□
A space tourism vehicle	□	□	□	□	□

A concurrent design philosophy integrates the different disciplines as early as possible into the design phase.

7. Concurrent engineering is the most accurate and efficient approach to design a space system.

□ Strongly Agree □ Agree □ Neutral □ Disagree □ Strongly Disagree

□ NA/Do not know

8. Please state if you think that there may be a better way (other than the concurrent one) to develop a space system. (Open question)

Multidisciplinary design optimization (MDO) is a field of engineering that uses optimization methods to solve design problems by simultaneously incorporating a number of disciplines and exploiting the interactions between them.

9. It is important to insert your domain of expertise in a MDO strategy.

□ Strongly Agree □ Agree □ Neutral □ Disagree □ Strongly Disagree

□ NA/Do not know

10. MDO has a great potential impact in the desgin of the total complex system.

□ Strongly Agree □ Agree □ Neutral □ Disagree □ Strongly Disagree

□ NA/Do not know

11. In which way do you see MDO being used during the design of space systems? (Open question)

12. What is your experience in the design of complex system in a cooperative environment?

□ More than 10 studies

□ 4-10 studies

□ 1-3 studies

□ Assisted to it / Have an idea

□ Never did it

□ NA/Do not know

Opinion on existing tools

13. Please explain which type of tools you have used as well as the context of their use.(Open question)

14. Please state the good and bad aspects of the integrated tools you have used. (Open question)

15. What is your overall impression of the integrated tools you have used

□ Very good and useful

□ Good but not so useful

□ Could be improved

□ Need to be improved

□ Very bad and inefficient

The design of the tool

In this section please give us your input for the design of a space systems integrated tool.

16. Please rank from 1 to 5 (1 being the most important) the importance of following aspects in the design of an integrated tool

	1	2	3	4	5
Representation of the overall system	☐	☐	☐	☐	☐
Integration of the different subsystems	☐	☐	☐	☐	☐
Optimization of design variables	☐	☐	☐	☐	☐
Real time update of design variables	☐	☐	☐	☐	☐
Short calculation times	☐	☐	☐	☐	☐

Representation of the overall system

17. It is convenient for the user to have a representation of the overall system.

☐ Strongly Agree ☐ Agree ☐ Neutral ☐ Disagree ☐ Strongly Disagree

☐ NA/Do not know

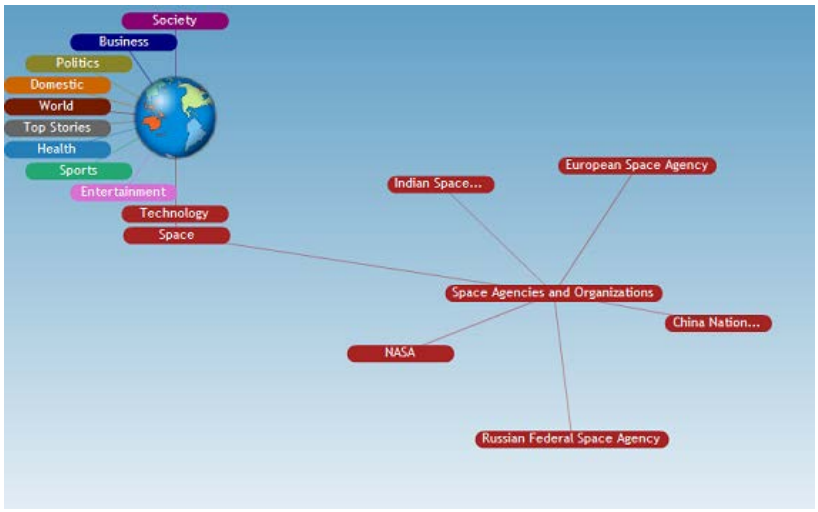


Figure 1: Example of dynamic representation levels (www.newstin.com)

18. It would be good to be able to choose at which level (abstraction layer) you want to visualize the system (in a way similar to the one in Figure 1)

☐ Strongly Agree ☐ Agree ☐ Neutral ☐ Disagree ☐ Strongly Disagree

☐ NA/Do not know

19. Please rank from 1 to 5 (1 being very important) the importance of including the following fields in an integrating tool

	1	2	3	4	5
Payload and Instrumentation	☐	☐	☐	☐	☐
Propulsion	☐	☐	☐	☐	☐
Spacecraft geometric configuration	☐	☐	☐	☐	☐
Structure	☐	☐	☐	☐	☐

Guidance Navigation & Control	☐	☐	☐	☐	☐
Thermal	☐	☐	☐	☐	☐
Power	☐	☐	☐	☐	☐
Communications	☐	☐	☐	☐	☐
Mechanisms	☐	☐	☐	☐	☐
Data Handling	☐	☐	☐	☐	☐
Radiation	☐	☐	☐	☐	☐
Mission Analysis	☐	☐	☐	☐	☐
Ground Segment and Operations	☐	☐	☐	☐	☐
Cost Analysis	☐	☐	☐	☐	☐
Risk Analysis	☐	☐	☐	☐	☐

Integration of different subsystems

20. The user must be allowed to have an idea of the impact of his/her design variables in the overall system

☐ Strongly Agree ☐ Agree ☐ Neutral ☐ Disagree ☐ Strongly Disagree

☐ NA/Do not know

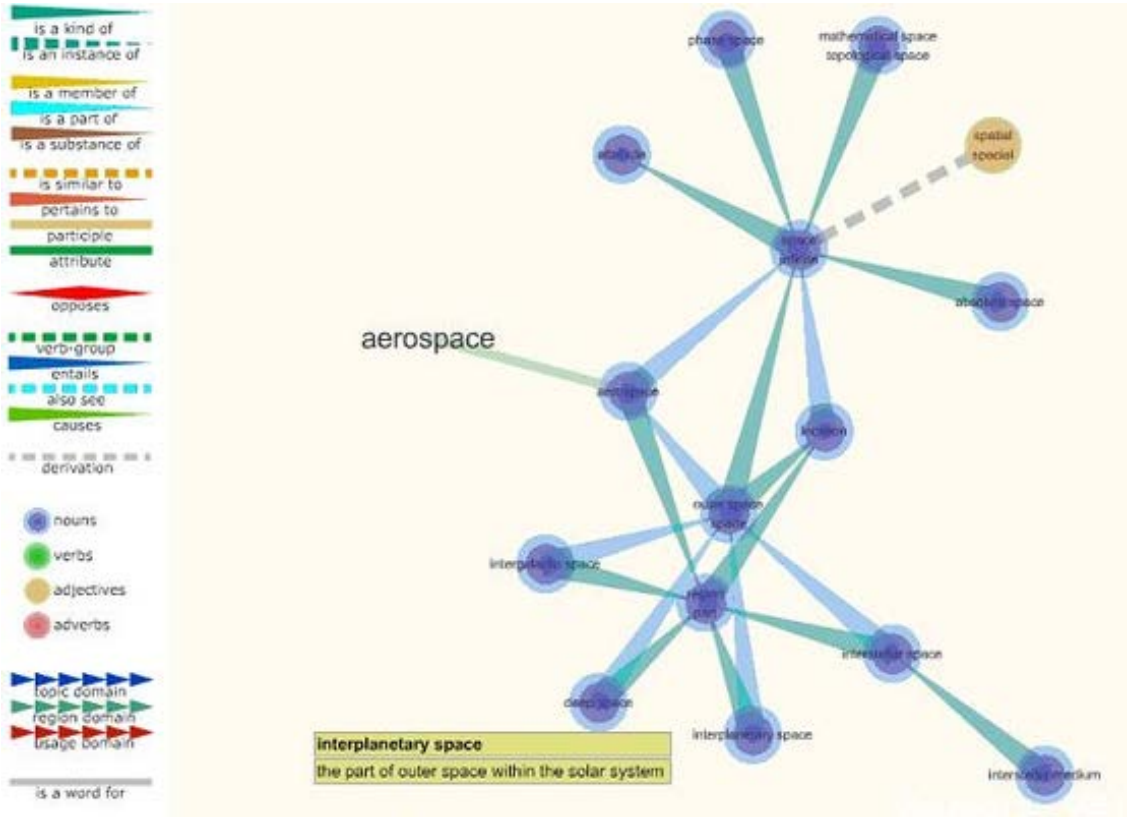


Figure 2: Example of a dependencies graph (http://www.visuwords.com)

21. The user must have access to the dependencies of his/her subsystem (like the one on Figure 2)

□ Strongly Agree □ Agree □ Neutral □ Disagree □ Strongly Disagree

□ NA/Do not know

22. Is there any type of visualization method that you know which you think would be useful on the design of space systems? (Open question)

Optimization of design variables

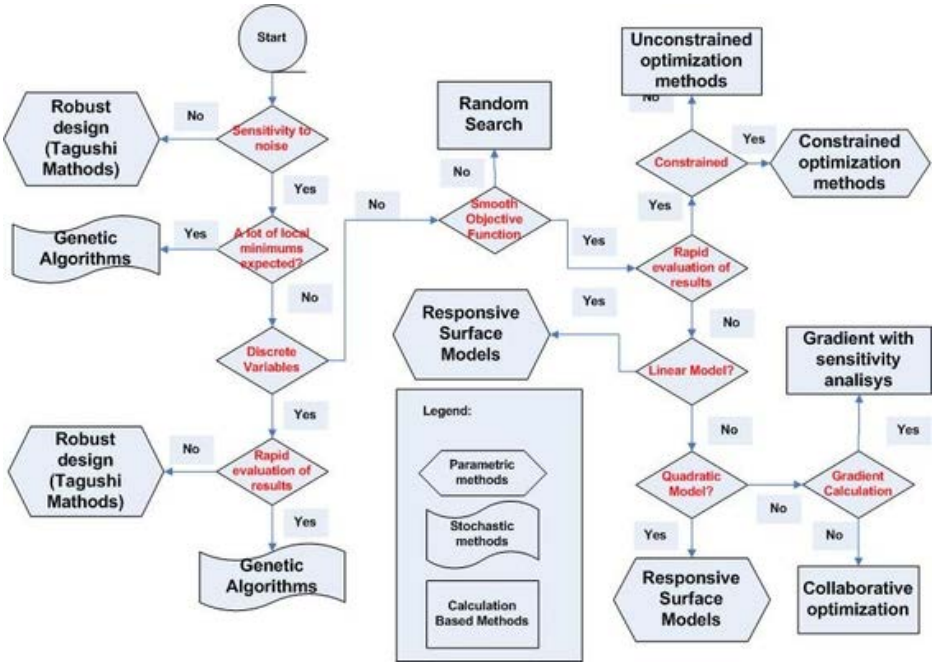


Figure 3: Choice of different optimization methods

23. From the earlier figure, please rank from 1 to 5 (1 being the most important) which of the optimization methods do you think would be more important to include in a space systems integrating tool?

	1	2	3	4	5
Robust Design Methods	□	□	□	□	□
Genetic Algorithms	□	□	□	□	□
Random Search	□	□	□	□	□
Collaborative Optimization	□	□	□	□	□
Gradient Methods	□	□	□	□	□

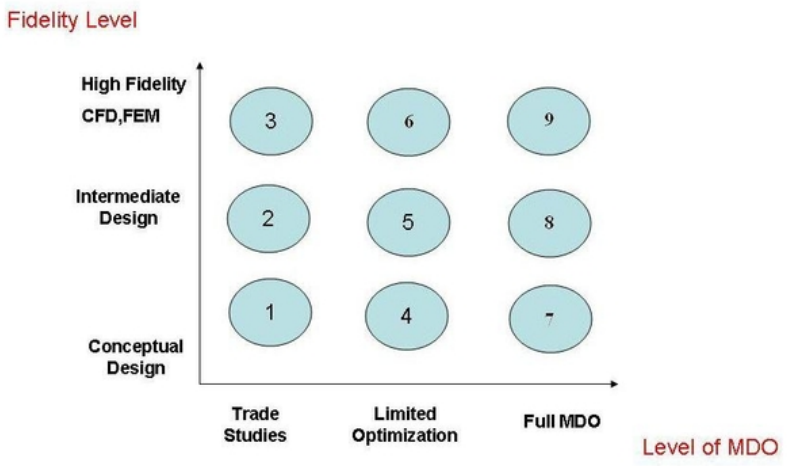


Figure 4: Different possibilities of MDO

24. Looking at the previous picture, in your opinion which type of MDO should a space systems integrating tool implement (in a preliminary design phase)

□ 1 □ 2 □ 3 □ 4 □ 5 □ 6 □ 7 □ 8 □ 9

25. How much time do you think it is acceptable to wait for the wait for the communication of the updated design variables (in seconds)?

Short calculation times

26. How much time do you think it is acceptable to wait for an overall design optimization calculation in the design phase (in minutes)?

[*This page intentionally left blank*]

10. APPENDIX 2

Performance tests with the predictive neural network-based decision support tool

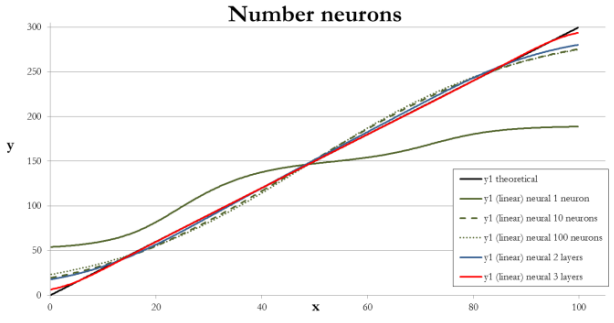


Figure 52: Influence of the neural network architecture on the predictions of y1(x) = 3x

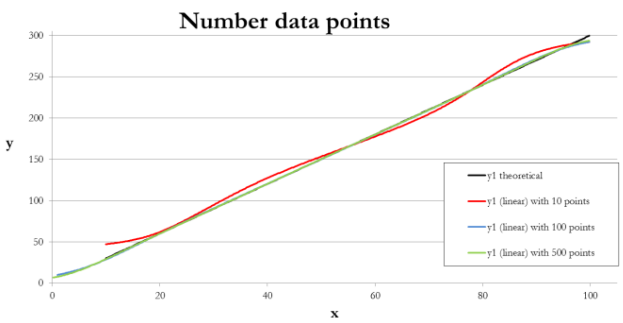


Figure 53: Influence of training dataset size on the predictions of y1(x) = 3x

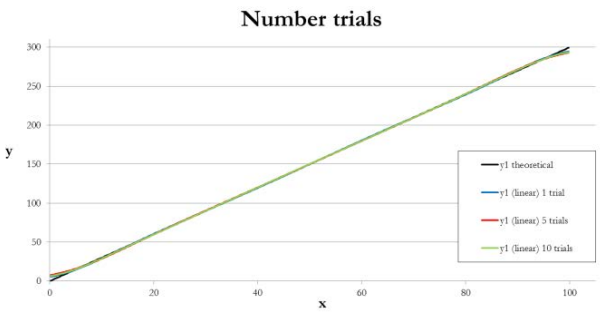


Figure 54: Influence of the number of trials on the predictions of y1(x) =

Figure 55: Influence of the error on the training dataset on the predictions of y1(x) = 3x

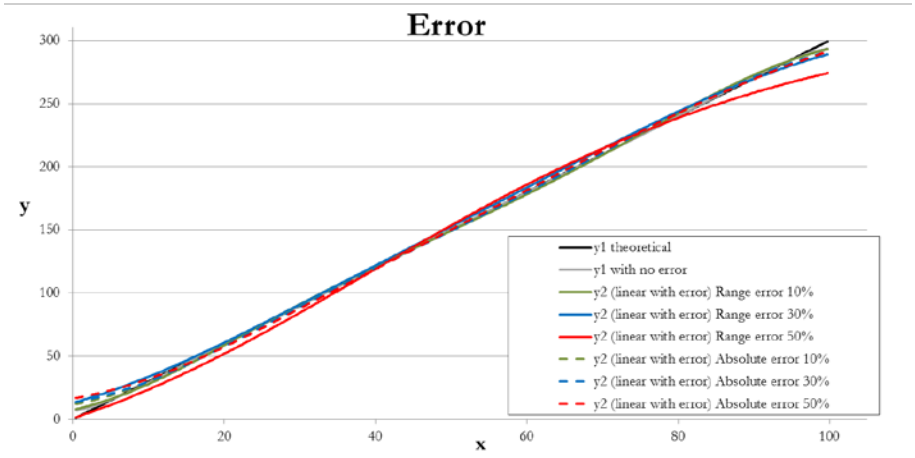
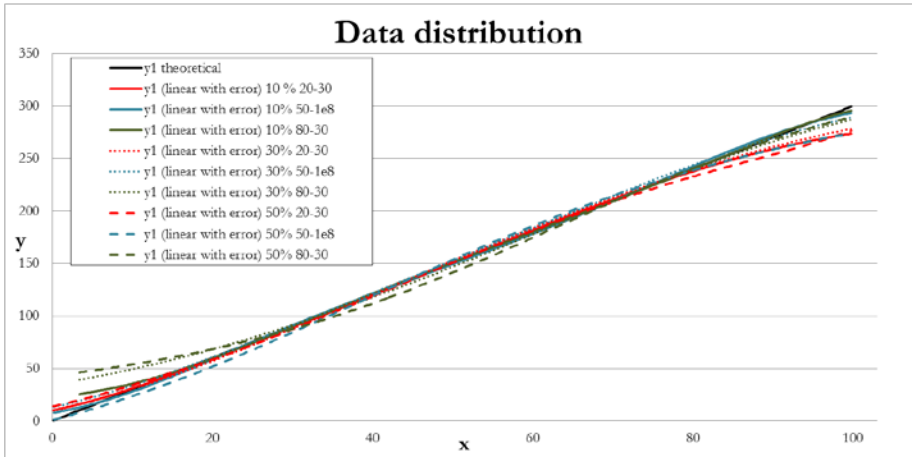


Figure 56: Influence of the training dataset distribution on the predictions of y1(x) = 3x



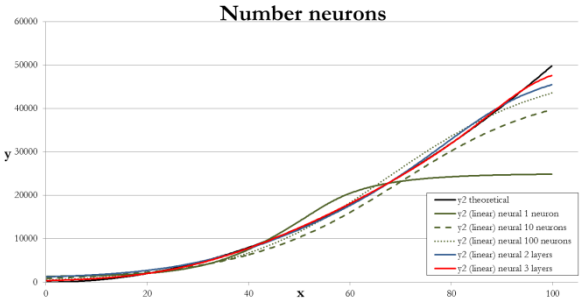


Figure 57: Influence of the neural network architecture on the predictions of y2(x) = x^2

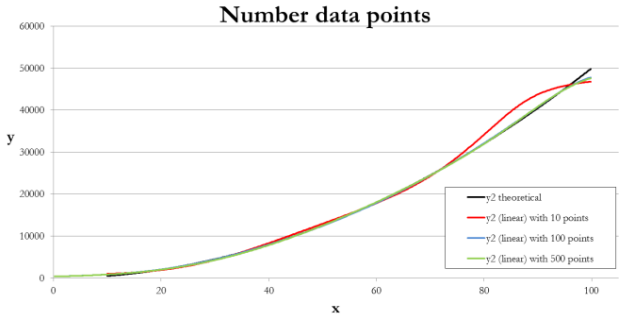


Figure 58: Influence of training dataset size on the predictions of y2(x) = x^2

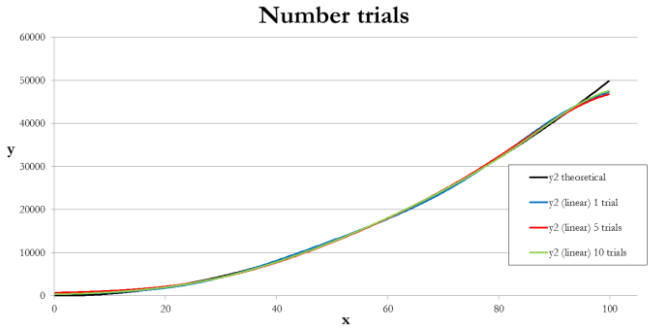
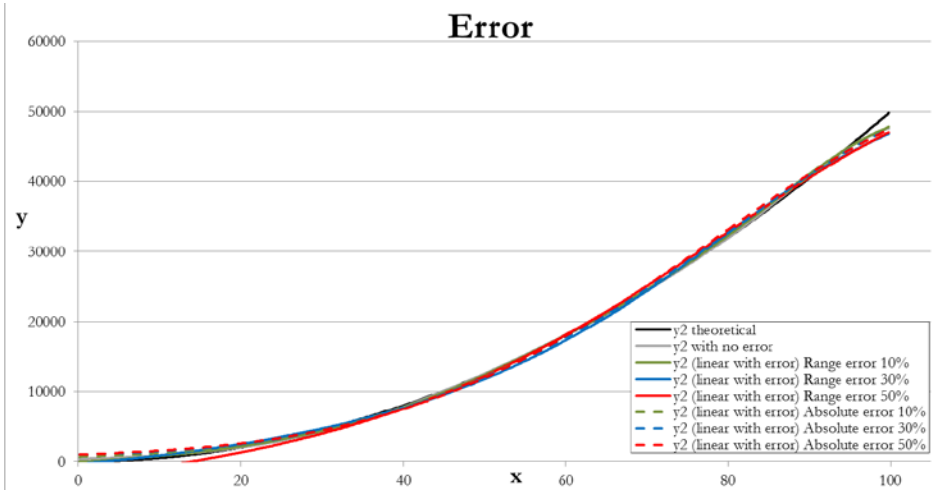
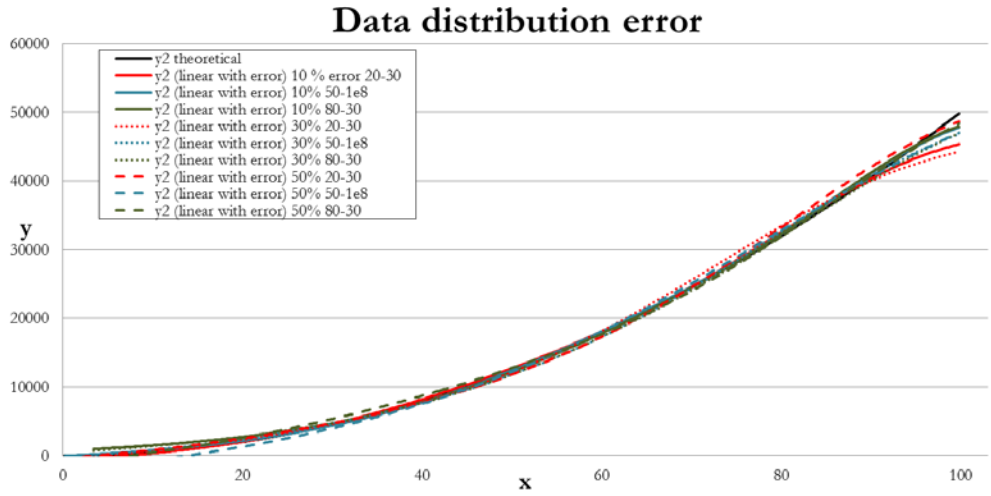


Figure 59: Influence of the number of trials on the predictions of y2(x) = x^2

Figure 60: Influence of the error on the training dataset on the predictions of y2(x) = x^2

Figure 61: Influence of the training dataset distribution on the predictions of y2(x) = x^2





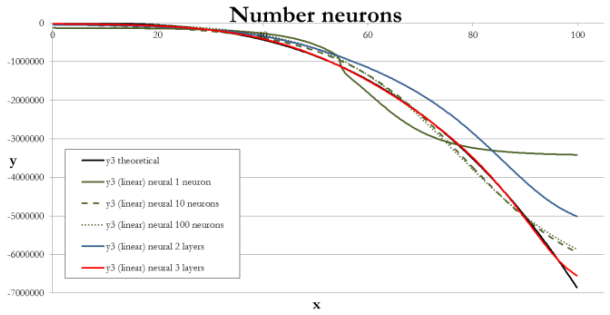


Figure 62: Influence of the neural network architecture on the predictions of y3(x) = -7*(x-0.5)^3

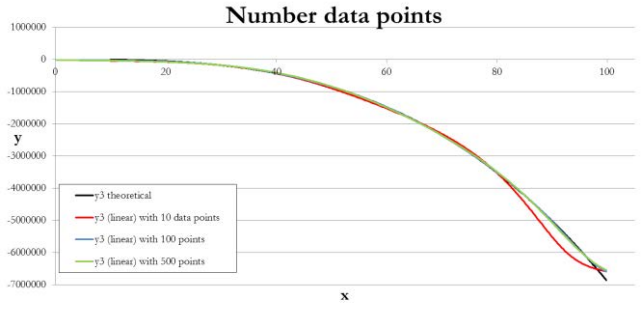


Figure 63: Influence of training dataset size on the predictions of y3(x) = -7*(x-0.5)^3

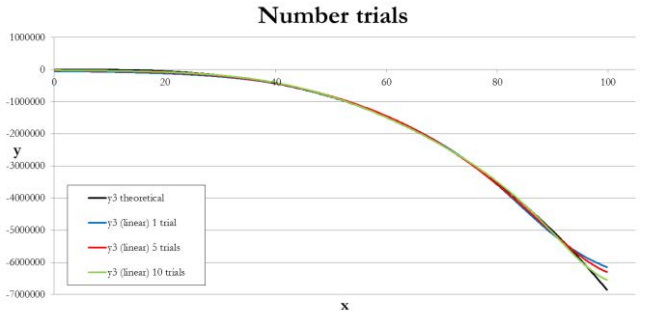


Figure 64: Influence of the number of trials on the predictions of y3(x) = -7*(x-0.5)^3

Figure 65: Influence of the error on the training dataset on the predictions of y3(x) = -7*(x-0.5)^3

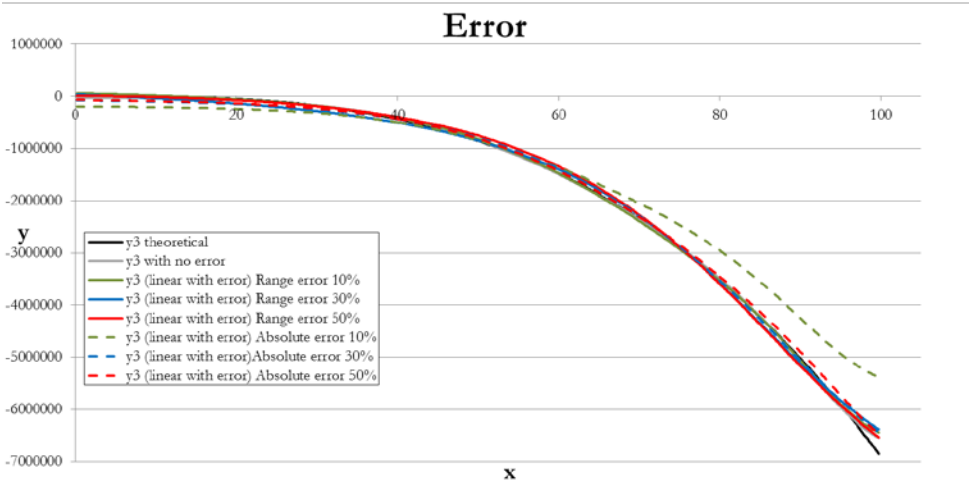


Figure 66: Influence of the training dataset distribution on the predictions of y3(x) = -7*(x-0.5)^3

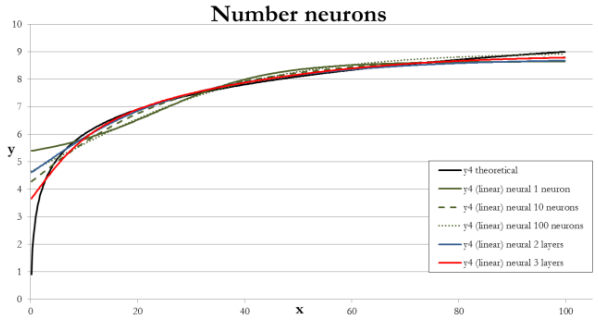


Figure 67: Influence of the neural network architecture on the predictions of y4(x) = 3*log(10x)

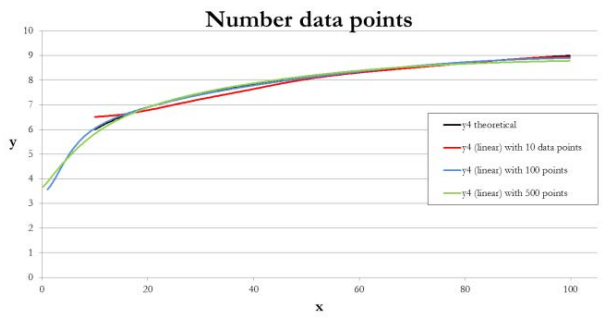


Figure 68: Influence of training dataset size on the predictions of y4(x) = 3*log(10x)

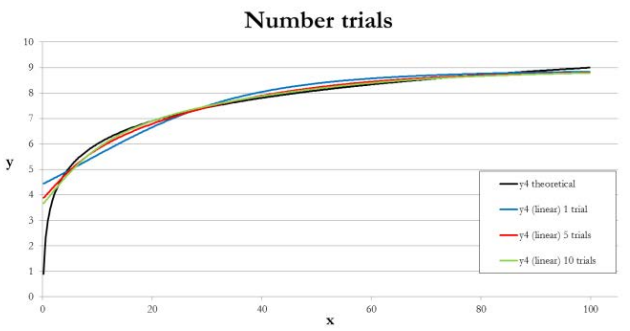


Figure 69: Influence of the number of trials on the predictions of y4(x) = 3*log(10x)

Figure 70: Influence of the error on the training dataset on the predictions of y4(x) = 3*log(10x)

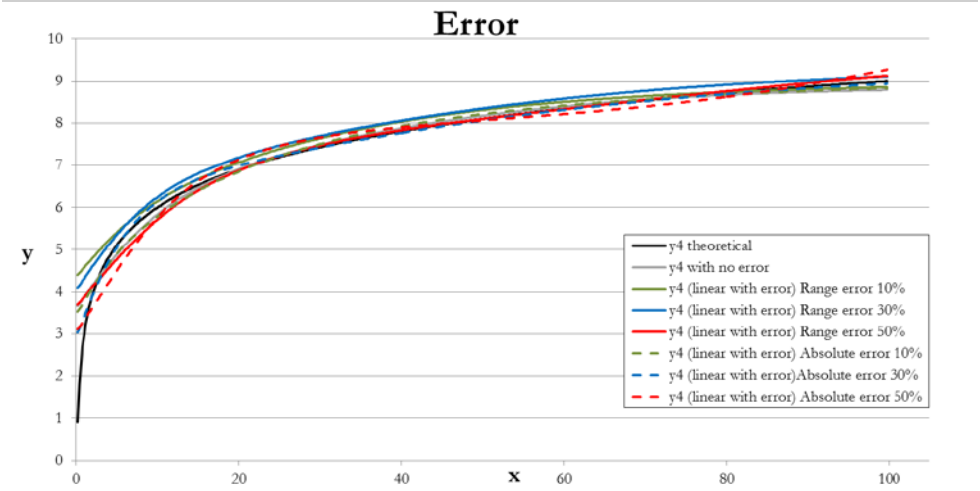
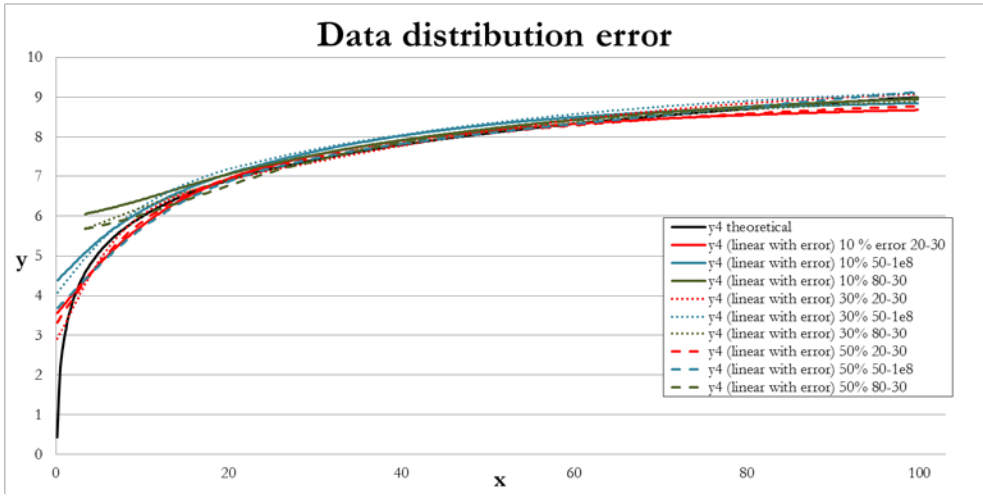


Figure 71: Influence of the training dataset distribution on the predictions of y4(x) = 3*log(10x)



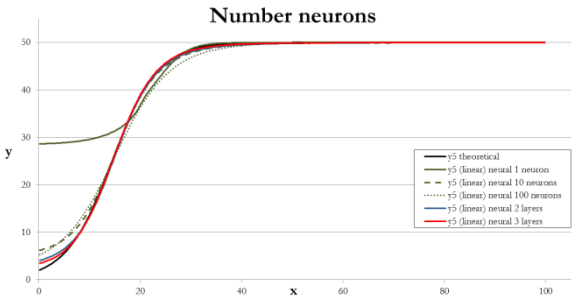


Figure 72: Influence of the neural network architecture on the predictions of y5(x) = 50/(1+72^(x/5))

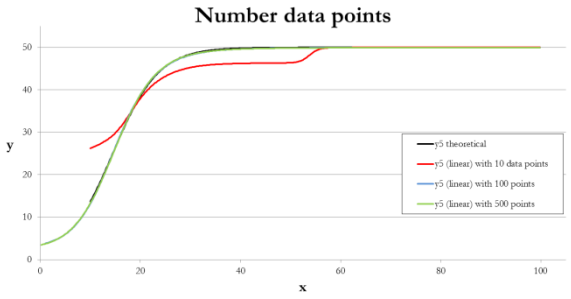


Figure 73: Influence of training dataset size on the predictions of y5(x) = 50/(1+72^(x/5))

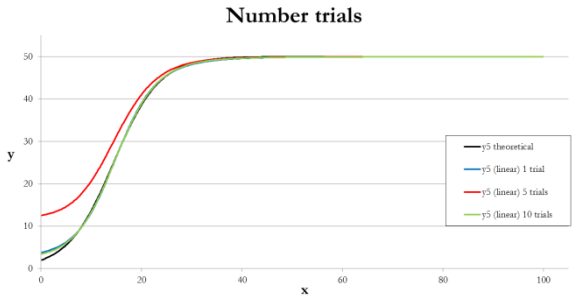


Figure 74: Influence of the number of trials on the predictions of y5(x) = 50/(1+72^(x/5))

Figure 75: Influence of the error on the training dataset on the predictions of y5(x) = 50/(1+72^(x/5))

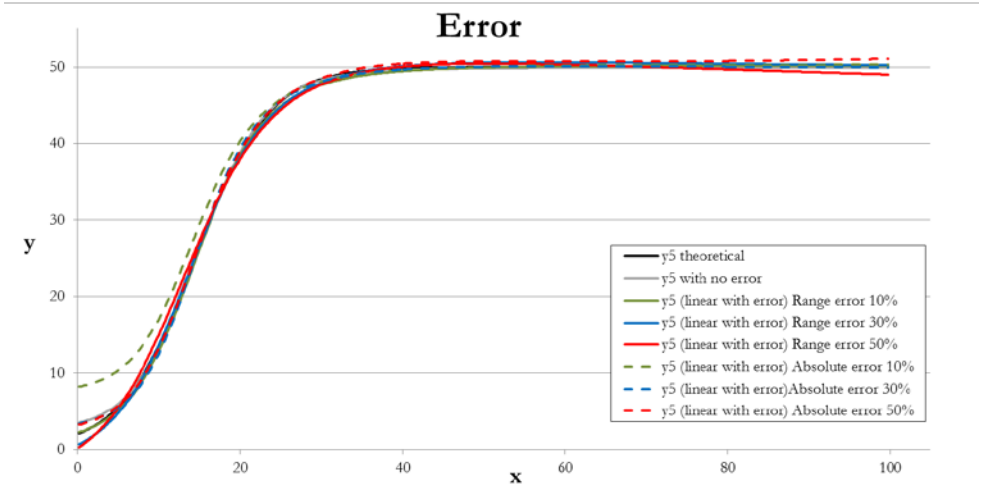
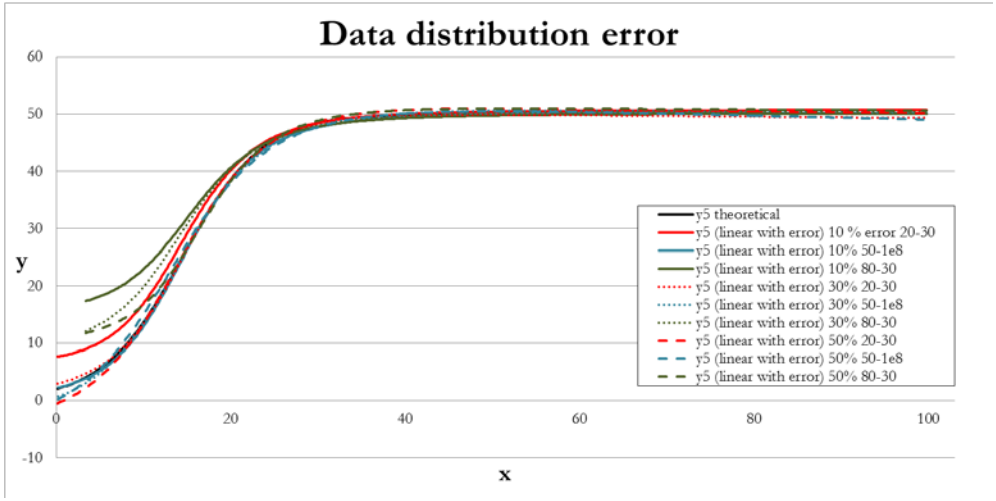


Figure 76: Influence of the training dataset distribution on the predictions of y5(x) = 50/(1+72^(x/5))



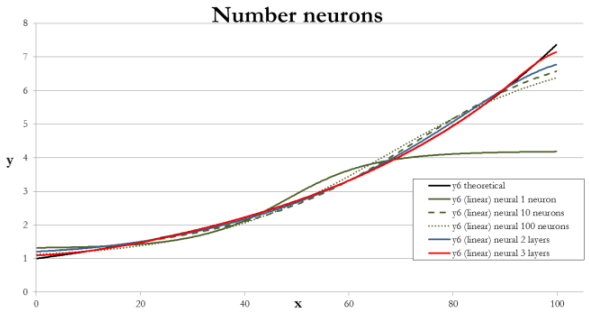


Figure 77: Influence of the neural network architecture on the predictions of y6(x) = exp(x/50)

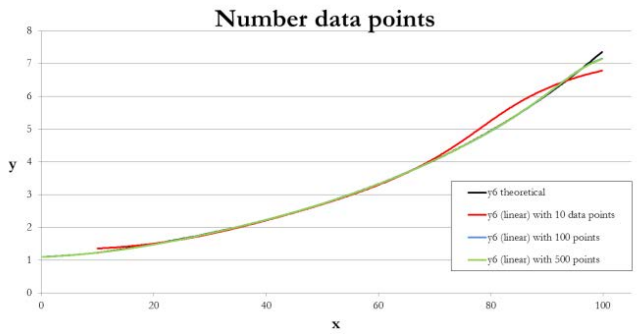


Figure 78: Influence of training dataset size on the predictions of y6(x) = exp(x/50)

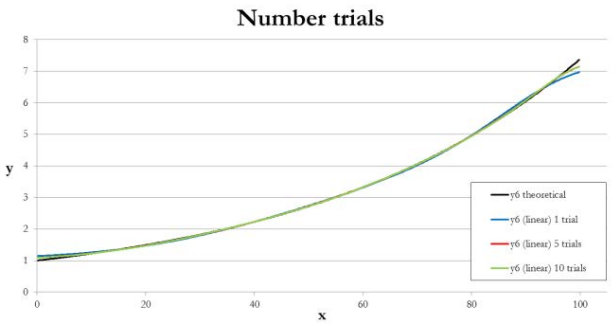


Figure 79: Influence of the number of trials on the predictions of y6(x) = exp(x/50)

Figure 80: Influence of the error on the training dataset on the predictions of y6(x) = exp(x/50)

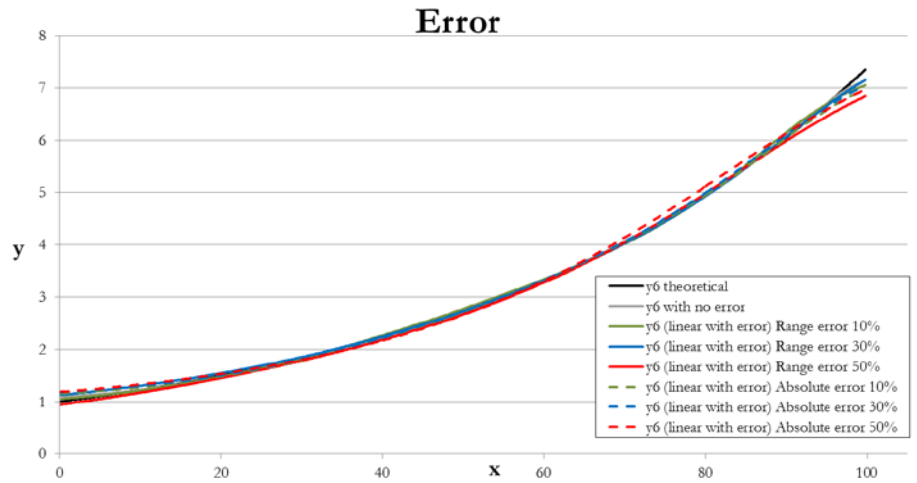


Figure 81: Influence of the training dataset distribution on the predictions of y6(x) = exp(x/50)

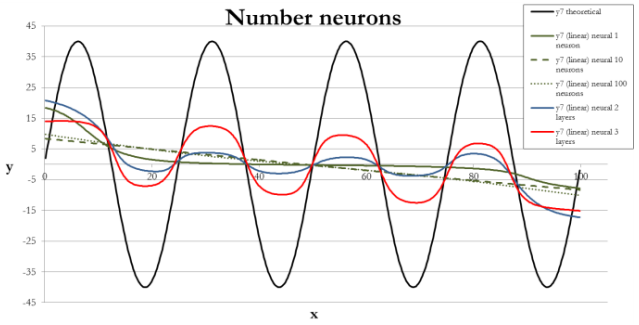


Figure 82: Influence of the neural network architecture on the predictions of y7(x) = 40*sin(8*pi*x/100)

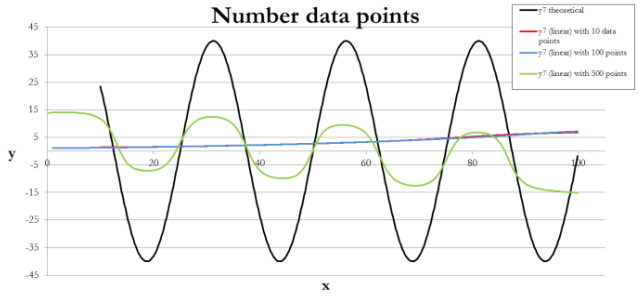


Figure 83: Influence of training dataset size on the predictions of y7(x) = 40*sin(8*pi*x/100)

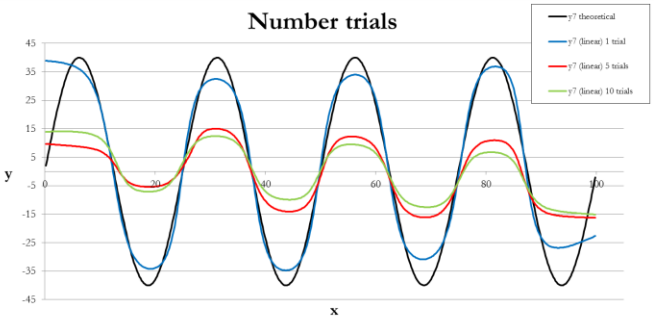


Figure 84: Influence of the number of trials on the predictions of y7(x) = 40*sin(8*pi*x/100)

Figure 85: Influence of the error on the training dataset on the predictions of y7(x) = 40*sin(8*pi*x/100)

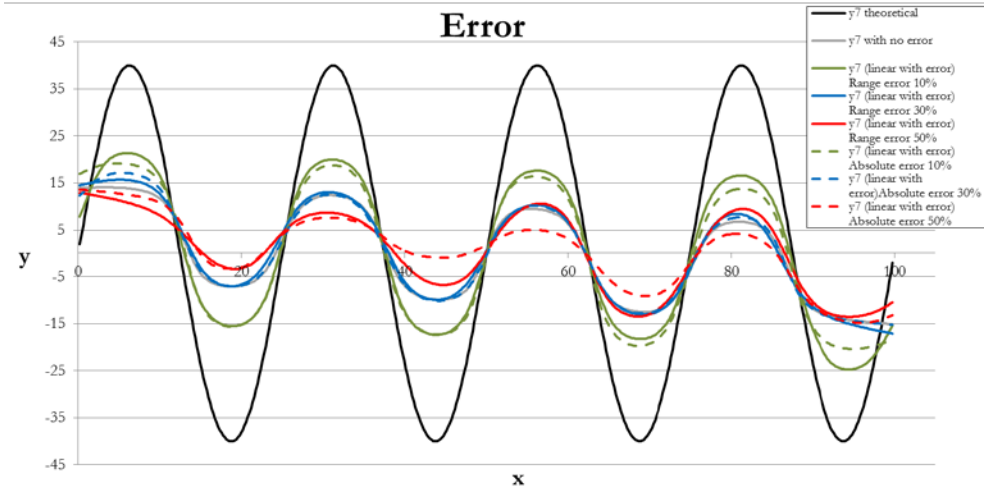
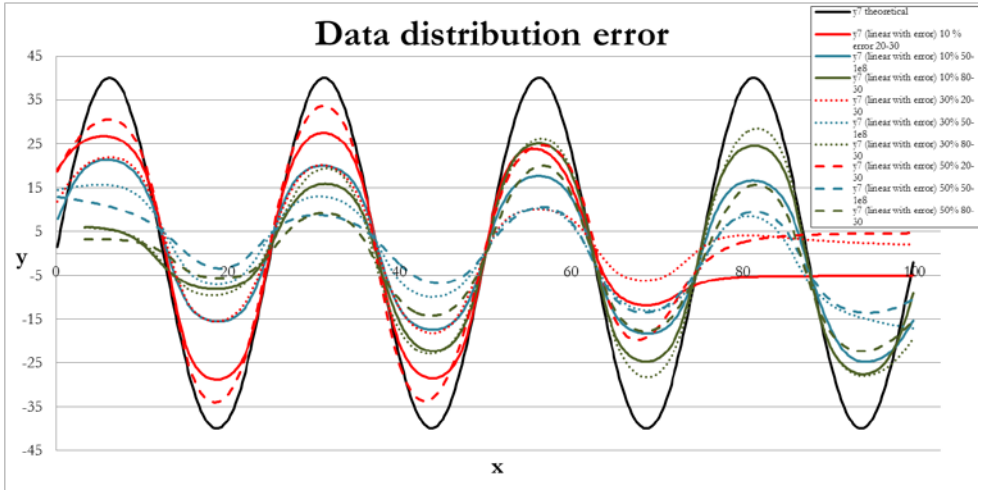


Figure 86: Influence of the training dataset distribution on the predictions of y7(x) = 40*sin(8*pi*x/100)



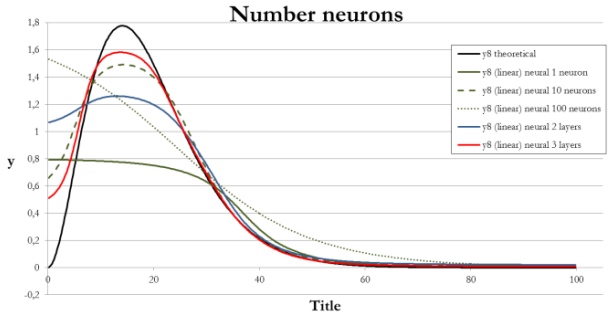


Figure 87: Influence of the neural network architecture on the predictions of y8(x) = 0.01*x^3/(exp(x/5)-1)

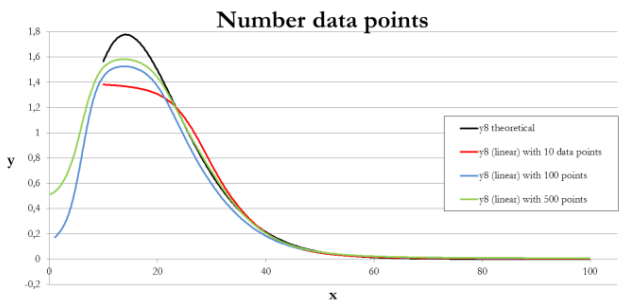


Figure 88: Influence of training dataset size on the predictions of y8(x) = 0.01*x^3/(exp(x/5)-1)

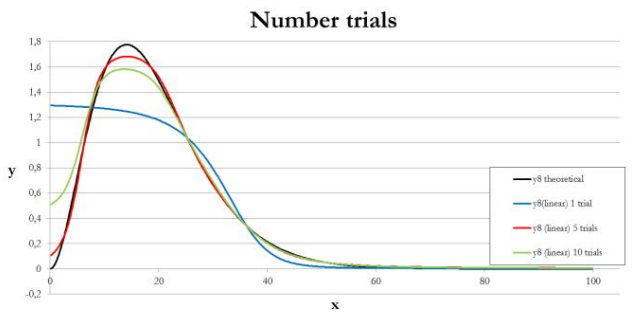


Figure 89: Influence of the number of trials on the predictions of y8(x) = 0.01*x^3/(exp(x/5)-1)

Figure 90: Influence of the error on the training dataset on the predictions of y8(x) = 0.01*x^3/(exp(x/5)-1)

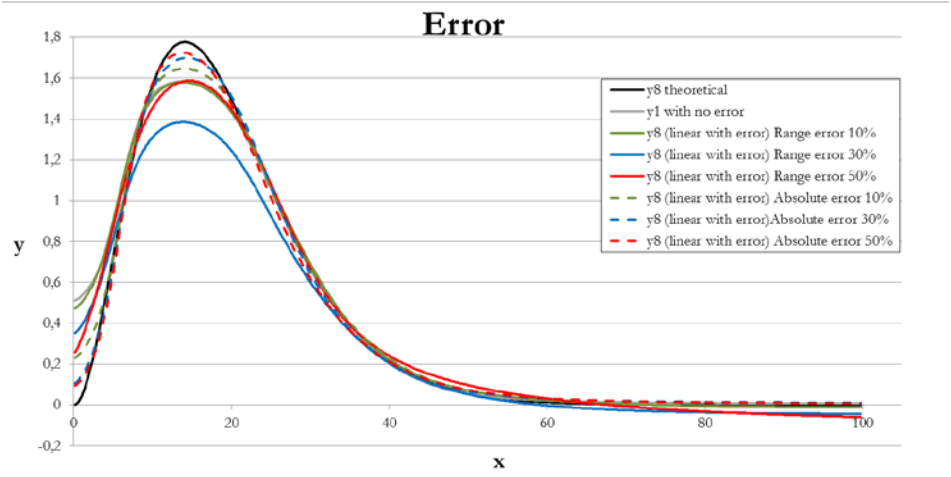
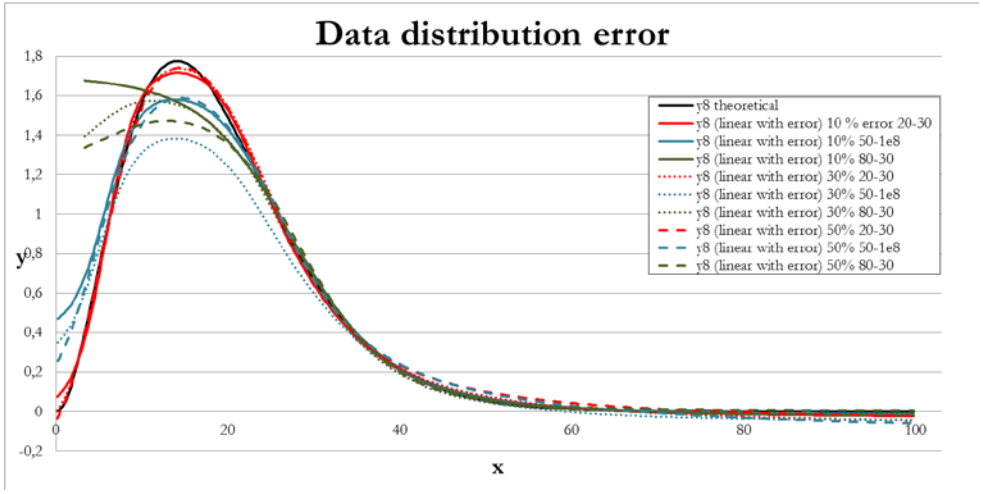


Figure 91: Influence of the training dataset distribution on the predictions of y8(x) = 0.01*x^3/(exp(x/5)-1)



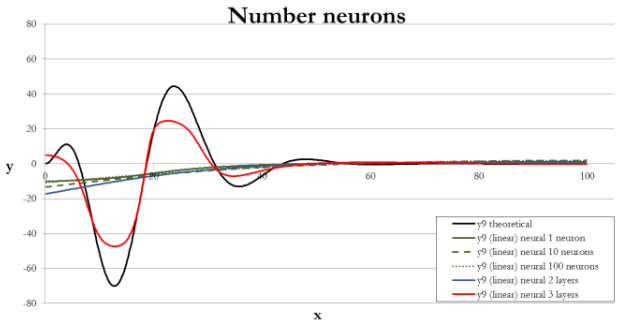


Figure 92: Influence of the neural network architecture on the predictions of y9(x) = (0.01*x^3/(exp(x/5)-1))* (40*sin(8*pi*x/100))

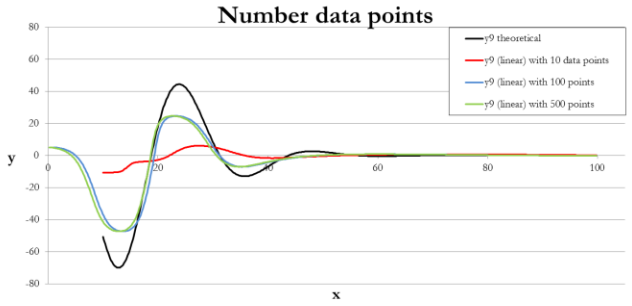


Figure 93: Influence of training dataset size on the predictions of y9(x) = (0.01*x^3/(exp(x/5)-1))* (40*sin(8*pi*x/100))

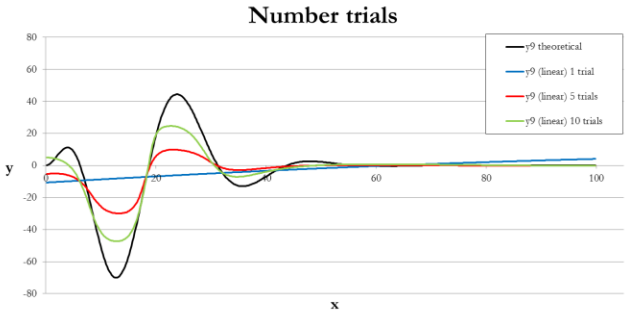


Figure 94: Influence of the number of trials on the predictions of y9(x) = (0.01*x^3/(exp(x/5)-1))* (40*sin(8*pi*x/100))

Figure 95: Influence of the error on the training dataset on the predictions of y9(x) = (0.01*x^3/(exp(x/5)-1))* (40*sin(8*pi*x/100))

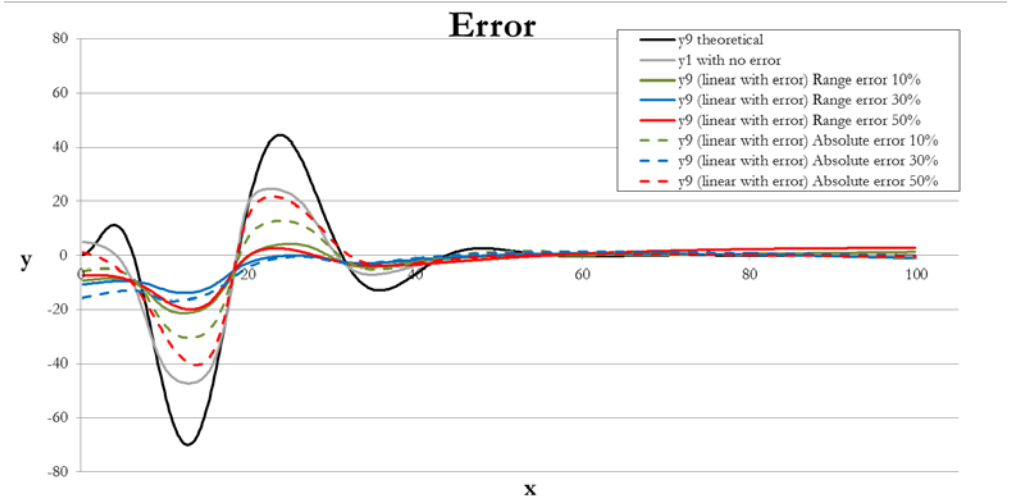
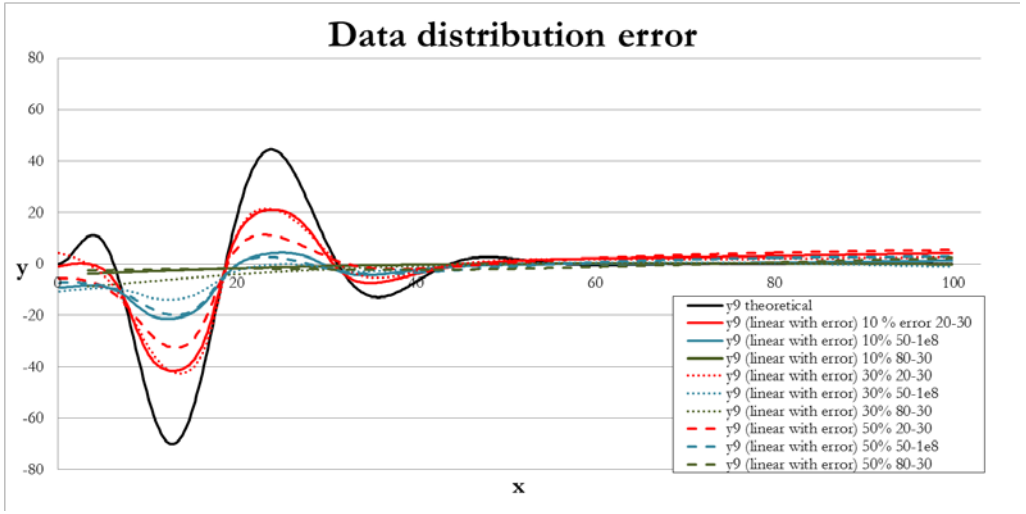


Figure 96: Influence of the training dataset distribution on the predictions of y9(x) = (0.01*x^3/(exp(x/5)-1))* (40*sin(8*pi*x/100))



[*This page intentionally left blank*]

APPENDIX 3

Example of handouts given during the case study «Thrust» with and without the use of the predictive decision support tool.

10.1. Case study «Thrust» (without tool)

Student identification number: _____

You are a propulsion specialist during a concurrent design session and you are responsible for the conceptual design of a new SSTO (Single Stage to Orbit) launcher. The structures subsystem is evaluating the loads during launch and is asking you for an estimation of the thrust for specific environmental conditions.

In your lab you have been doing a series of tests with similar thrusters in different environmental conditions but, unfortunately, you do not have the specific one to you are being requested. Your idea is to use the existing values that you obtained during your previous tests to predict the value of thrust for the conditions indicated by the structures subsystem: Ae (Exit nozzle section) – 3 m2, Pe (Exit nozzle pressure) – 0.3 bar, Pa (Atmospheric pressure) – 0.25 bar , Pc (Pressure at the combustion chamber) – 50 bar , At (Throat section) – 0.2 m2.

Input: Ae (Exit nozzle section) , Pe (Exit nozzle pressure), Pa (Atmospheric pressure), Pc (Pressure at the combustion chamber), At (Throat section)

Output: Thrust

Input Values: 3 m2, 0.3 bar, 0.25 bar, 50 bar, 0.2 m2

Database: Thrust_Database

Results:

Which method did you use to reach your solution:
- ☐ 1. Approximation model
- ☐ 2. Specific Model
- ☐ 3. Guess
- ☐ 4. Other

In case you chose 1, 2, ou 4 detail how you created the model used:

If applicable, which type of interpolation or approximation method did you use (e.g. 3rd degree polynomial with r^2=0.965)

_____

_____

_____

Why?

_____

_____

How did you prepare your data to be used in your model (e.g. filtered data from x to y and then only considered even):

_____

_____

If applicable, what is the definition of the model you considered more (e.g. y= x^2+56)

_____

_____

Your suggestion: _____

Error estimation (%):_____

10.2. Case study «Thrust» (with tool)

Student identification number: _____

You are a propulsion specialist during a concurrent design session and you are responsible for the conceptual design of a new SSTO (Single Stage to Orbit) launcher. The structures subsystem is evaluating the loads during launch and is asking you for an estimation of the thrust for specific environmental conditions.

In your lab you have been doing a series of tests with similar thrusters in different environmental conditions but, unfortunately, you do not have the specific one to you are being requested. Your idea is to use the existing values that you obtained during your

previous tests to predict the value of thrust for the conditions indicated by the structures subsystem: Ae (Exit nozzle section) – 3 m2, Pe (Exit nozzle pressure) – 0.3 bar, Pa (Atmospheric pressure) – 0.25 bar , Pc (Pressure at the combustion chamber) – 50 bar , At (Throat section) – 0.2 m2.

Input: Ae (Exit nozzle section), Pe (Exit nozzle pressure), Pa (Atmospheric pressure), Pc (Pressure at the combustion chamber), At (Throat section)

Output: Thrust

Input Values: 3 m2, 0.3 bar, 0.25 bar, 50 bar, 0.2 m2

Database: Thrust_Database

Results:

Stopping criteria used:

- ☐ Time (How much?: _____)
- ☐ Training error (How much?: _____)
- ☐ Testing error (How much?: _____)
- ☐ Training or Testing error (How much?: _____)
- ☐ Automatic divergence detection

Number of runs: _____

Neuron distribution per layer: _____

Did you filter the training set? (Yes/No) : _____

If yes, what was the condition used?: _____

Your suggestion: _____

Error estimation (%):_____

[*This page intentionally left blank*]

11. APPENDIX 4

This appendix details the main functionalities of INCREMENT and introduces some of the support software technologies that enable its operation.

INCREMENT software description

The core of INCREMENT is a complex relational database in MySQL with a total of 38 tables to monitor elements such as parameters, design items (which can represent either system states or physical parts), disciplines, justifications, and parameter assignments (Figure 66).

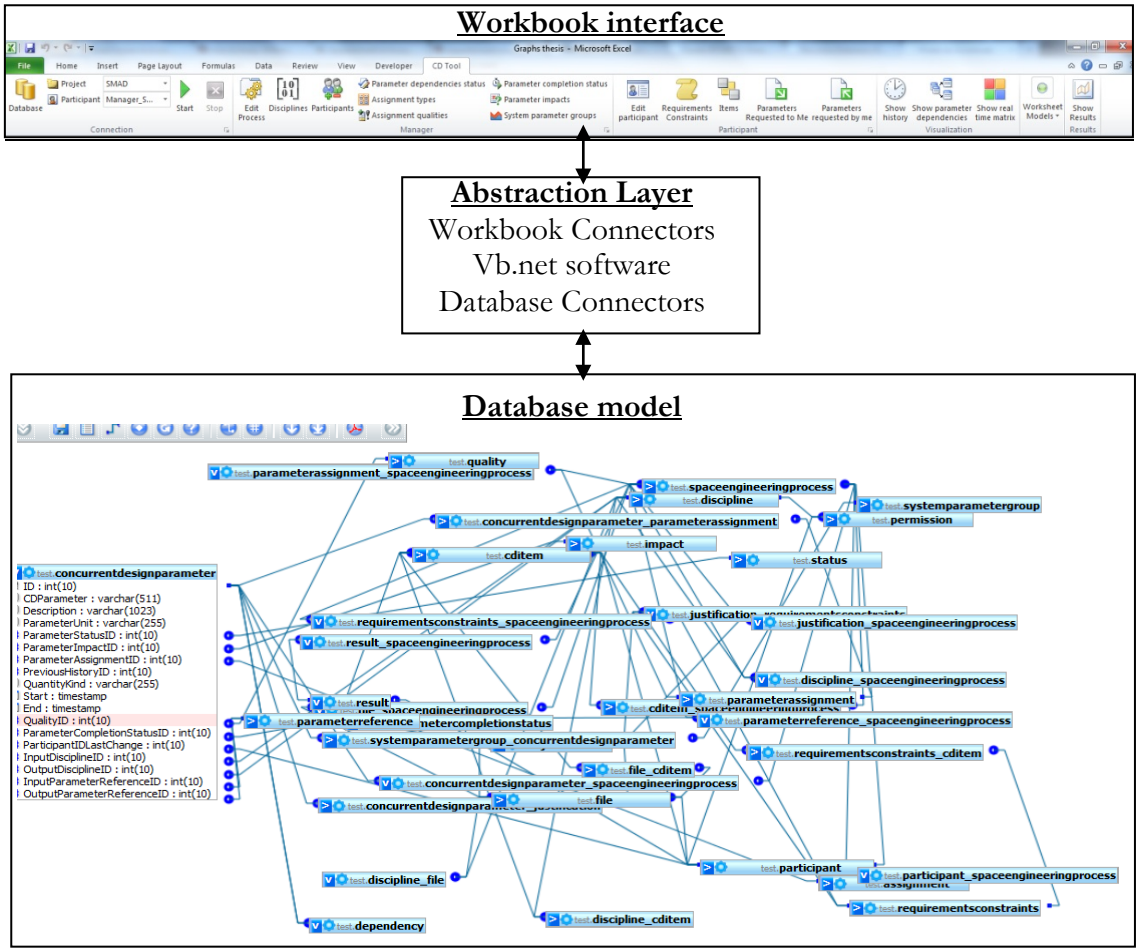


Figure 66: INCREMENT architecture

In order to reduce the required learning curve, the interface is a ribbon on a common workbook editor. In the same platform, designers are able to define the subsystem level models, manage all the interactions, document every design change, and have a global view of the system.

11.1. Setup connection and interaction mode

The first menu of INCREMENT sets up a connection with the database. The database can be placed in a local server, a dedicated server over the internet, or as a service through cloud computing service-providers.

Once the connection is established, the designer can either load an existing project by confirming his/her username and password or, create a new system by specifying its characteristics such as Name, Lifecycle Stage, Activity Phase, Number of CE Session and password of the System Manager.

When a new system is created, only the system manager has access to the environment before adding new participants or disciplines through the manager functionalities menu.

11.2. Manager functionalities

Some of the functionalities of INCREMENT can only be accessed by the manager of the system, who has total control over all the disciplines, models, parameters and design items. This is a role commonly taken by project managers or system engineers in the context of CE environments.

The manager is able to define different disciplines (also known as subsystems or domains), different participants, and their respective permissions. Each designer can be assigned read and/or write permissions to be able to access and/or change design models, parameters and items (Figure 67).

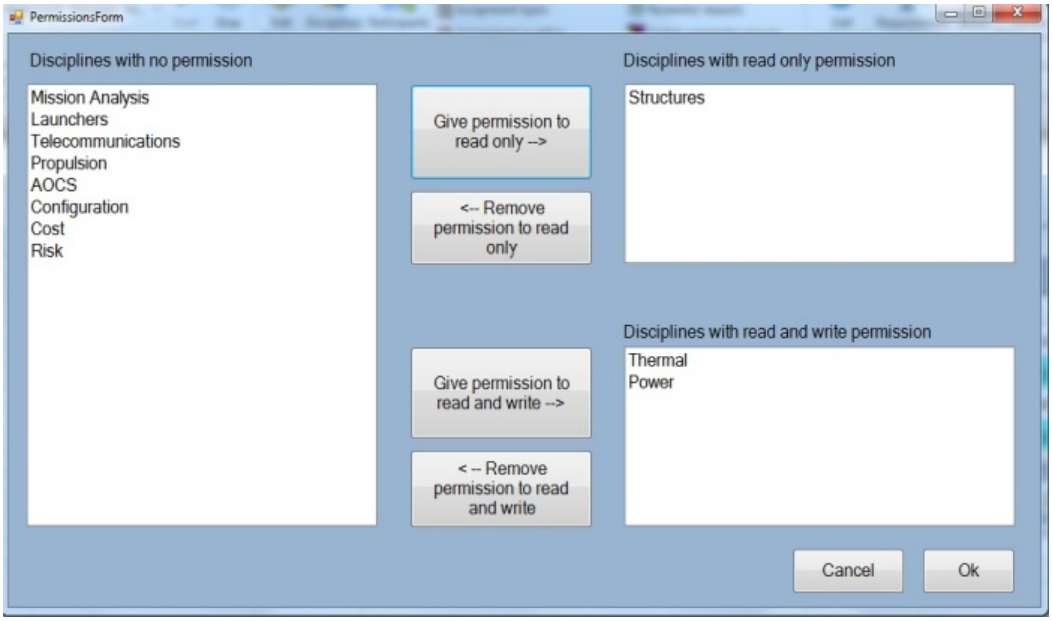


Figure 67: Definition of permissions

11.3. Participant functionalities

The participant functionalities are common to every designer, whether they are managers or subsystem specialists. The options in this menu allow the definition of new requirements or constraints and control of design items. The designers are also able to create new design parameters (request them), or change the attributes of existing ones, with respect to status of the relationship, assignments, type of assignment, quality of the assignment, parameter impact, or parameter completion status (§4.3) (Figure 68).

Once the designer does any change to any of the parameters he/she has the possibility of including a justification which can be a simple text, attachment to a file, or a link to a previously defined requirement or constraint.

If the designer enables the automatic update mode (synchronous collaboration), the different worksheets give an overview of the current status of the parameters requested to other disciplines, parameters requested to his/her discipline (or the one to which he/she has write permissions), parameter assignments, design items, and requirements or constraints that need to be taken into account when designing his/her discipline.

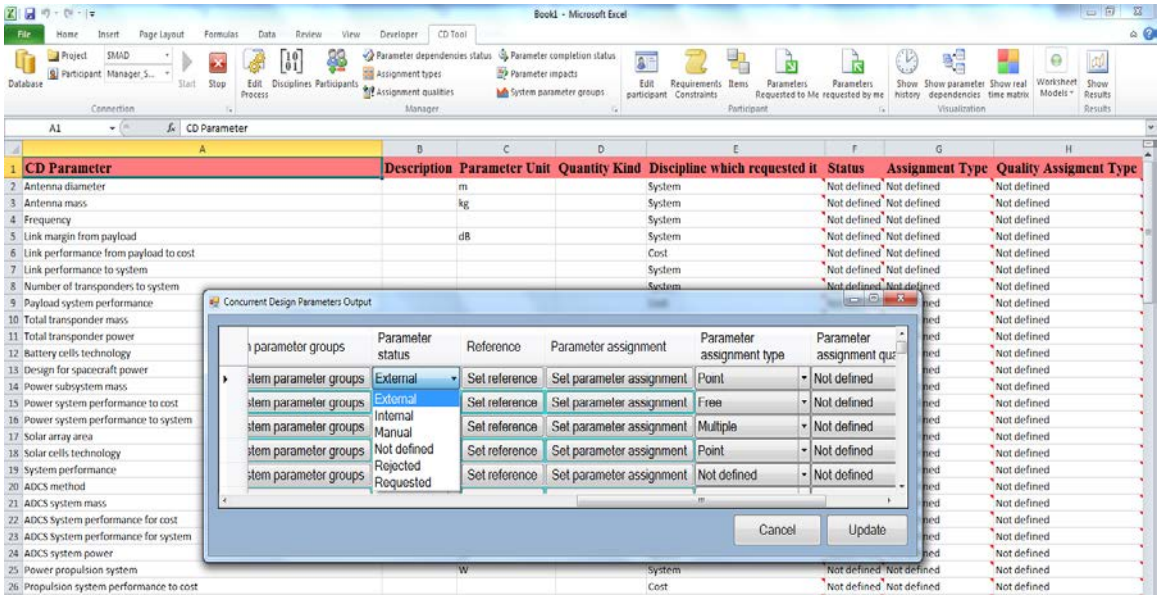


Figure 68: Changing the status of a requested design parameter through the participant functionalities menu for an Earth-observation satellite. Through an intuitive interface, the designer is able to change the current status of a parameter by choosing amongst the possible states defined by the manager. In this example the designer is able to change the parameter dependency status to: external, internal, manual, rejected or requested (as described in §4.3.1). If the design is on the synchronous mode, the changes will immediately be available to all the other subsystems.

11.4. Visualization functionalities

The visualization functionalities allow a real-time view of the current status of the design process as well as a historical assessment of its evolution.

When choosing the «DSM-view» detailed in section 4.3, the designer is able to assign a visual metaphor (color, font size, font color, extra text, or pattern), up to a maximum of 5 simultaneous codes, to the different issues to track (status of the relationship, parameter assignment type, quality of assignment, parameter impact, and completion status). The DSM is then generated and will be updated real-time whenever one of these characteristics changes for any of the design parameters. This visualization gives an instantaneous view of the overall status of the design process, in terms of the status of relationships, types of assignments, impact of parameters, or completion status of a design model. It is a valuable view to increase the perception of the coupling between the different design parameters, enabling the detection of possible design bottlenecks and increasing the insight into the behavior of the overall system (Figure 69).

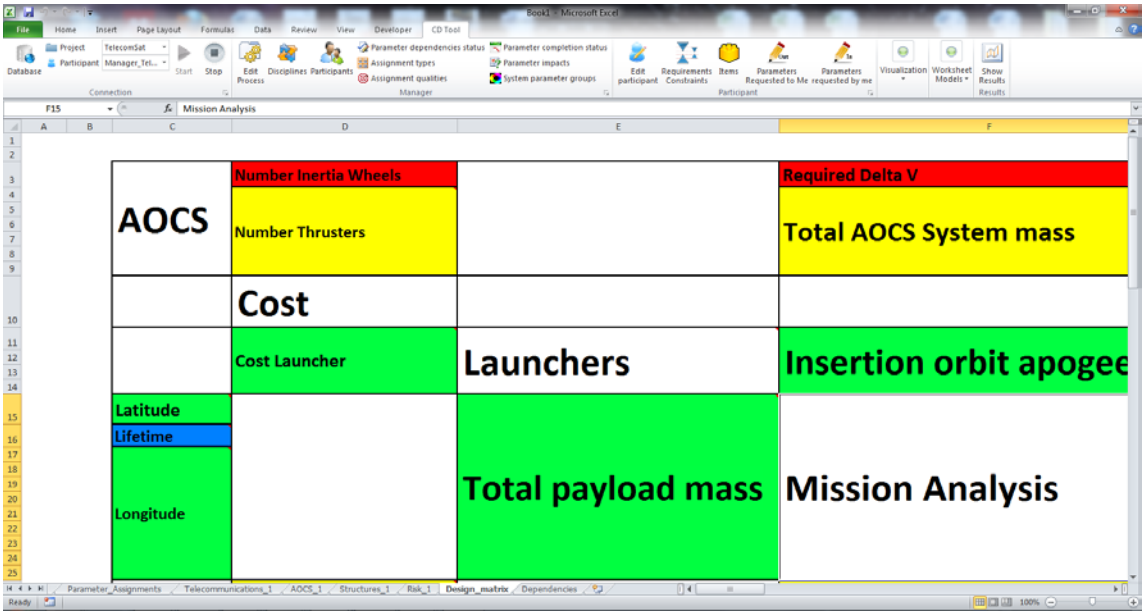


Figure 69: Part of the «DSM view» of an Earth-observation satellite. In this example, the different colors represent different status of the relationship (red= dependency not accepted, yellow= dependency accepted but not defined, green = dependency accepted, blue = dependency with an underlying model), different font size represents the different types of assignment (discrete, continuous, single point)

The ability to track parameter dependencies was defined as a very helpful visualization method during the survey performed to experienced users (Chapter 3) and included in the earlier methodology (Chapter 4). INCREMENT implements this functionality within the visualizations menu by enabling the designer to choose between dependencies or precedencies of a specific design parameter. This visualization is automatically updated

whenever any discipline changes its model that links to the design parameter of any of the cells defined by a function. It provides a viewpoint on the information flow up to a certain depth (defined by the user) and informs the designer of the last changes with their respective justifications (Figure 70).

Finally, the designer can choose to display the historical changes of any of the design parameters, design items or design models. Depending on the permissions, there is a possibility to revert any of these changes and, therefore, return to an earlier point in time.

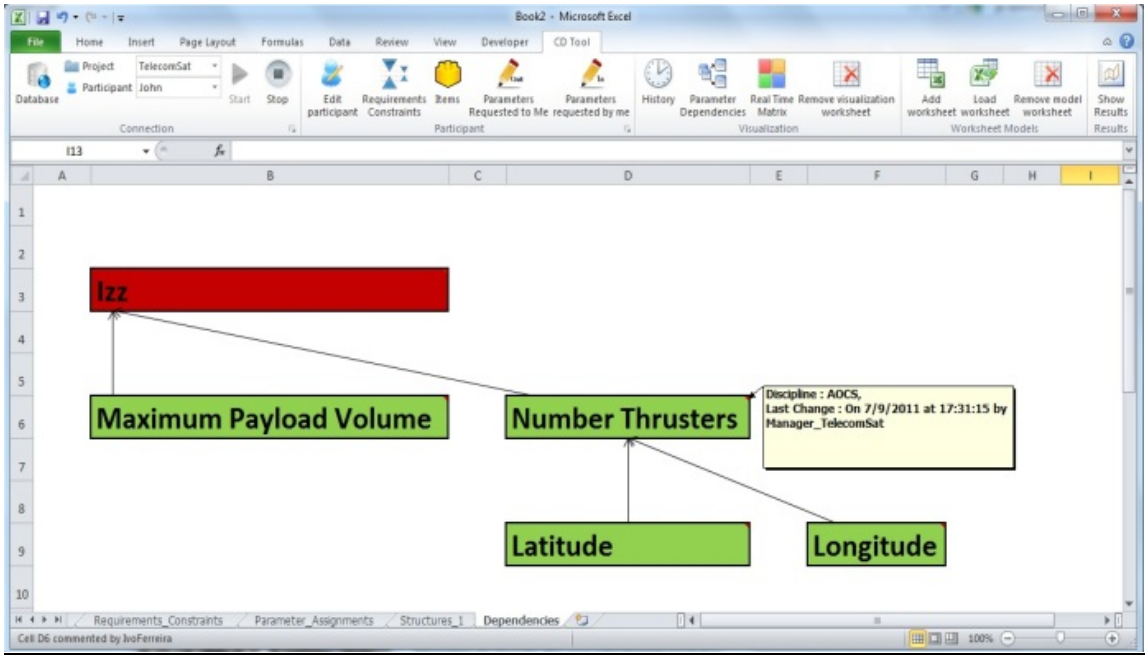


Figure 70: Example of the dependencies visualization (up to the second degree) for the «Izz» parameter defined by the structures discipline of a telecommunication satellite. This visualization is automatically updated whenever the subsystem models change. In this case, the parameter «Izz», depends on the parameters: «Maximum payload volume» and «Number of thruster» defined by the AOCS subsystem.

11.5. Subsystem modeling functionalities

Another advantage of INCREMENT lies in the fact of using common workbooks with which the majority of the designers are already acquainted. Using the worksheet modeling capabilities bypasses the need of establishing another modeling language and can reduce the learning curve. Each designer is able to create new worksheets to define the models that support each of the design parameters (Figure 71).

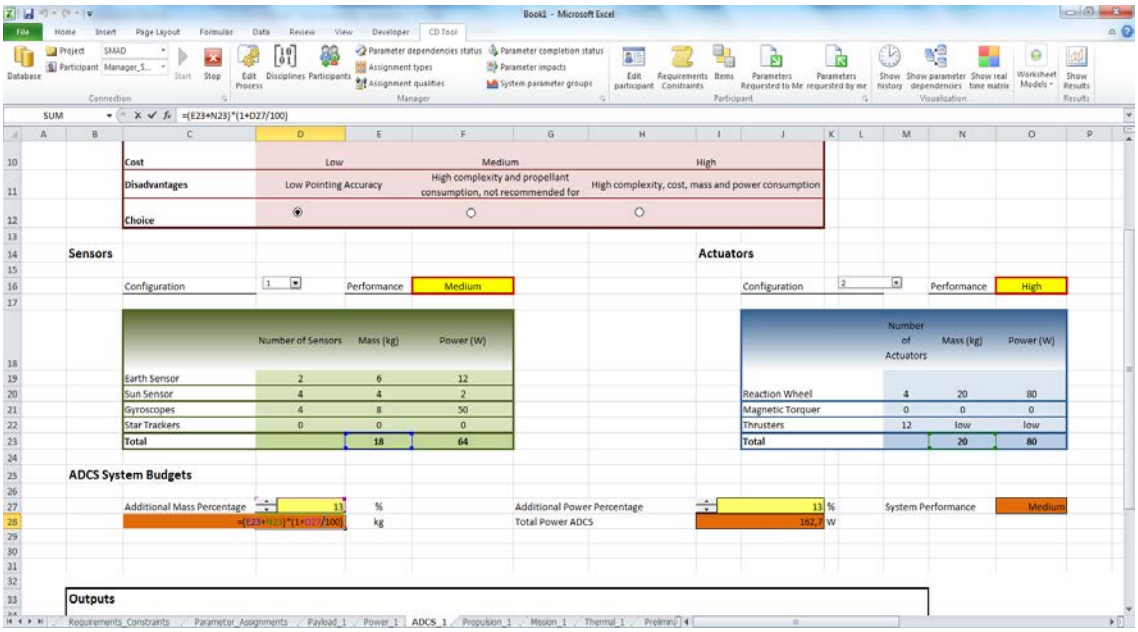


Figure 71: Model definition for one of the subsystems of a Earth-observation satellite (Attitude Control System). All the relations between parameters can be defined through simple worksheets.

By defining cell references for the parameter inputs and outputs, the models can be fully integrated. Parameters that do not have an underlying model can be assigned a fixed value. More complex models can be created with more detailed tools and then loaded into the worksheet, using lookup tables, or with the help of data wrappers designed for that specific application. All the design changes are changed at all steps to the database (Figure 72).

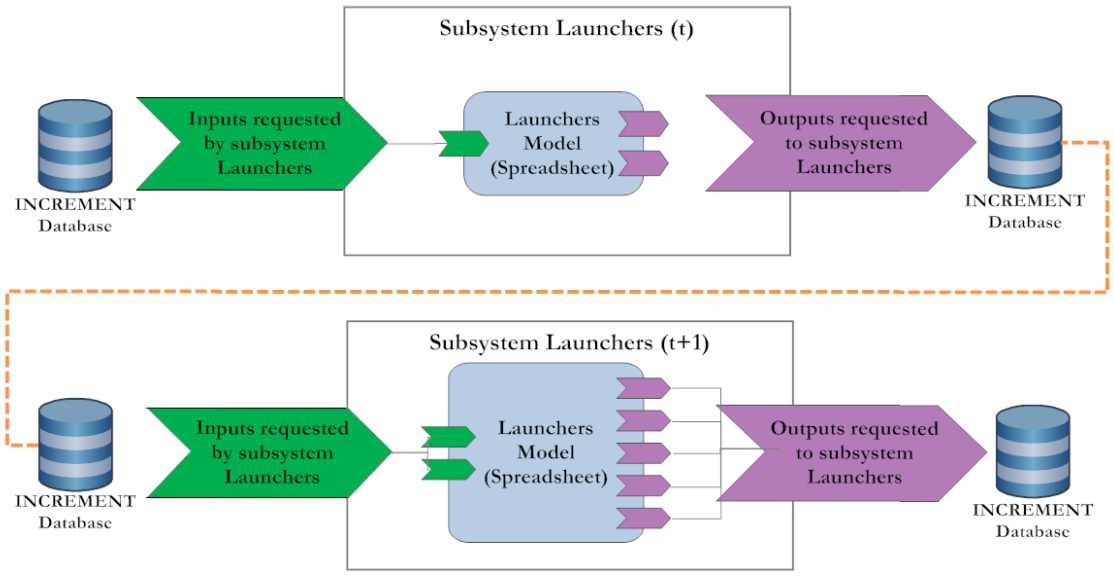


Figure 72: Linking the subsystem models to INCREMENT. It is possible to change the evolution of the design model and update its connection to INCREMENT throughout the design process.

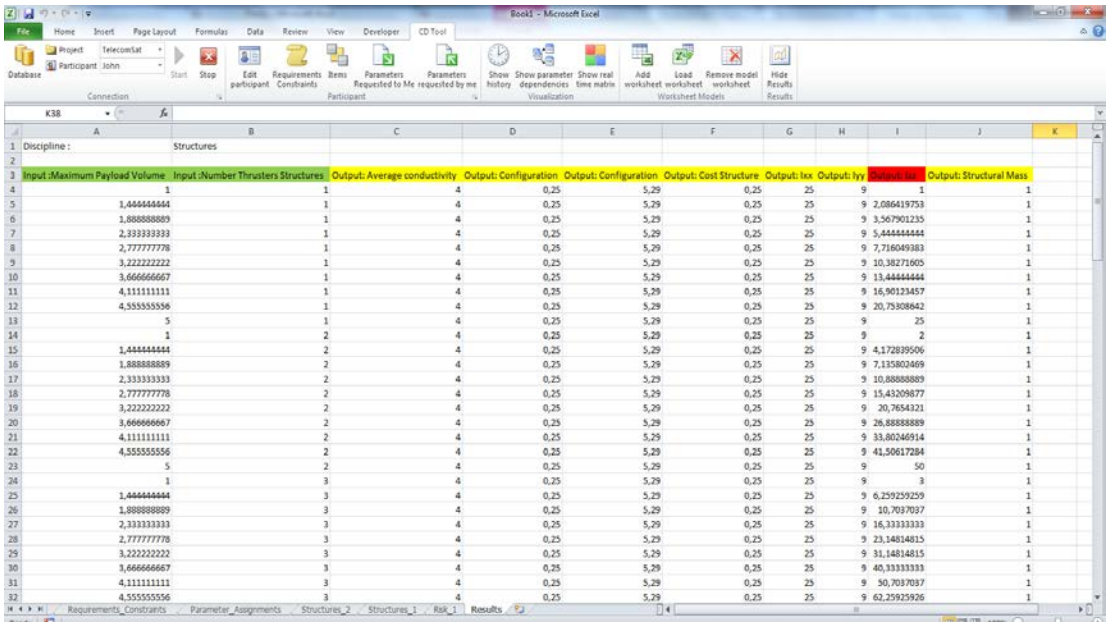
As the worksheets are loaded into the database, the designers are asked for a justification for the changes and, at that point, all the dependencies and possible results are recalculated.

The designer will always be able to go back to early design models by reverting previous changes.

11.6. Generation of design possibilities

The functionalities described in the previous sections correspond to instruments that can support most of the stages of the integrated design methodology proposed in Chapter 4, from the general description stage to the end of the interaction stage.

Once the design parameters are defined and the subsystems level models linked, INCREMENT can use the assignments made to the input parameters of each subsystem to calculate the different design possibilities based on the models that were defined at the subsystem level. These results can be the departure point for a tradespace exploration strategy (G. Stump et al. 2009) (§4.4). By choosing the «generate possible results» option, the designer can be presented with a list of the possible designs according to the subsystem level models, and the parameter assignments defined by the upstream disciplines (Figure 73).



Discipline :	Structures								
Input :Maximum Payload Volume	Input :Number Thrusters Structures	Output: Average conductivity	Output: Configuration	Output: Configuration	Output: Cost Structure	Output: Ixx	Output: Iyy	Output: Izz	Output: Structural Mass
1	1	4	0,25	5,29	0,25	25	9	1	1
1,444444444	1	4	0,25	5,29	0,25	25	9	2,086419753	1
1,888888889	1	4	0,25	5,29	0,25	25	9	3,567901235	1
2,333333333	1	4	0,25	5,29	0,25	25	9	5,444444444	1
2,777777778	1	4	0,25	5,29	0,25	25	9	7,716049383	1
3,222222222	1	4	0,25	5,29	0,25	25	9	10,38271605	1
3,666666667	1	4	0,25	5,29	0,25	25	9	13,44444444	1
4,111111111	1	4	0,25	5,29	0,25	25	9	16,90123457	1
4,555555556	1	4	0,25	5,29	0,25	25	9	20,75308642	1
5	1	4	0,25	5,29	0,25	25	9	25	1
1	2	4	0,25	5,29	0,25	25	9	2	1
1,444444444	2	4	0,25	5,29	0,25	25	9	4,172839506	1
1,888888889	2	4	0,25	5,29	0,25	25	9	7,135802469	1
2,333333333	2	4	0,25	5,29	0,25	25	9	10,88888889	1
2,777777778	2	4	0,25	5,29	0,25	25	9	15,43209877	1
3,222222222	2	4	0,25	5,29	0,25	25	9	20,7654321	1
3,666666667	2	4	0,25	5,29	0,25	25	9	26,88888889	1
4,111111111	2	4	0,25	5,29	0,25	25	9	33,80246914	1
4,555555556	2	4	0,25	5,29	0,25	25	9	41,50617284	1
5	2	4	0,25	5,29	0,25	25	9	50	1
1	3	4	0,25	5,29	0,25	25	9	3	1
1,444444444	3	4	0,25	5,29	0,25	25	9	6,259259259	1
1,888888889	3	4	0,25	5,29	0,25	25	9	10,7037037	1
2,333333333	3	4	0,25	5,29	0,25	25	9	16,33333333	1
2,777777778	3	4	0,25	5,29	0,25	25	9	23,14814815	1
3,222222222	3	4	0,25	5,29	0,25	25	9	31,14814815	1
3,666666667	3	4	0,25	5,29	0,25	25	9	40,33333333	1
4,111111111	3	4	0,25	5,29	0,25	25	9	50,7037037	1
4,555555556	3	4	0,25	5,29	0,25	25	9	62,25925926	1

Figure 73: Example of the possible results for one of the outputs of the risk discipline in the context of a telecommunications satellite. These design possibilities are automatically generated in function of the parameter assignments, if the designers define ranges for the possible values (or lists), the feasibility is calculated to every the different possibilities. In this example, every row is a different feasible design.

The design possibilities are updated whenever any of the upstream subsystems (that define the input design parameters) change its parameter assignments.

With INCREMENT, the design process can converge from an initial set of possible results to a final one through a set of judicious decisions or more elaborate MDO strategies.