Architecture decisions in different product classes for complex products

Marija Jankovic, Claudia Eckert

Architecture decisions in different product classes for complex products

Marija Jankovic,

Laboratoire de Génie Industriel

CentraleSupélec, Université de Paris Saclay

Grande Voie des Vignes, 92290 Chatenay Malabry

Phone: +33 1 41 13 14 27

e-mail: marija.jankovic@centralesupelec.fr

Claudia Eckert

Department of Engineering and Innovation

The Open University

Walton Hall, Milton Keynes,

MK7 6AA United Kingdom

e-mail: claudia.eckert@open.ac.uk

Abstract:

Many of the most fundamental decision about a product are taken during the system architecture design process. However how system architecture is designed in practice is not well understood. This paper draws on several research studies related to system architecture design to develop a categorization of system architecture design processes to support the adaptation design methodologies and tools to specific situations. The paper reviews different definitions of system architecture and comments on the relevance of the different perspectives taken in the literature on system architecture to different types of system architecture. The research highlights the need for further empirical research on system architecture design processes as well as on tools to support the engineers creating the system architecture.

Key words: System Architecture, Decision making, Complex system

1 INTRODUCTION

Most of the fundamental decisions about a product, which commit a large fraction of the cost and determine the success of the product in the market, are made at the very beginning of the design processes when system architecture is defined. At this stage, the companies largely finalise the requirements and establish the functional and the physical configuration of the product; and thereby also determine the tasks required in the new product development process. Yet this is process is not well understood in the academic literature and or supported by academic tools and methods. While much of the academic research was focussed on system architecture of novel designs, industry is developing complex projects in an incremental way and carries components and solution principles across to other products. Drawing on literature and the authors past case studies this paper therefore set up to identify different categories of system architecture design and discusses how these affect the system architecture process.

There are several definitions of system architecture. Ulrich and Eppinger (1995) define the system architecture as "the arrangement of the functional elements into physical blocks". Ulrich (1995) refines this definition further "(1) the arrangement of functional elements; (2) the mapping from functional elements to physical components; (3) the specification of the interfaces among interacting physical components". Crawley (2007) defines the system architecture as "the embodiment of concept and the allocation of physical/informational function to elements of form, and definition of interfaces among elements and with the surrounding context".

While much design research has been carried out into the creation of design concepts through experimental studies or generative systems and modelling methods for both requirements and functional modelling, the process of designing the system architecture of complex products in industry is not yet clearly identified or understood. Based on our own case studies and those published in the literature (Chepko, De Weck et al. 2008, Bonjour, Deniaud et al. 2009, Albers, Braun et al. 2011, Moullec, Bouissou et al. 2013) this paper will argue that there is a huge variation in the process of creating a system architecture and the decisions required to define a system architecture; and therefore also the support that designers require to do so. The system architecture of a highly innovative one-off product, such as a space shuttle where new technology is employed to meet newly identified functions is very different to that of a mass product incremental products like traditional cars. Observations from

industry underline the need for different design process and methods in designing system architectures with regard to these differences in products. In this paper, we provide a classification of different types of system architecture design problems in terms of the characteristics of the products and the contexts and constraints under which the design process takes place. We distinguish between the rare examples of (1) ab initio design; (2) incremental design where significant parts of the previous solution are carried over; (3) the reuse of solution principles where the technology is known but not the components; (4)platform design where the components or sub systems are shared across multiple products which intertwines the system architecture of multiple products; and (5) design for future flexibility which caters for future uncertainties. We also discuss the implication the different classes have for the system architecture process.

All the studies addressed complex engineered products, in the sense of the definition of Bloebaum and McGowan (2010). Bloebaum and McGowan define complex systems as "systems that have tightly coupled interaction as well as often unpredicted and emerging behaviour". The focus of this paper is on products with a significant mechanical / physical product components, rather than the architecture of non-physical systems, like service systems or the design of systems of systems, such as transport systems or airports. Many of the current generation of complex products require the integration of several design domains (mechanical, electronics, software engineering, product design, etc.). Each domain traditionally has its own approach to a domain system architecture design. The products are also deployed in a range of use contexts under a large number of different conditions. The products also have behavior that is hard to predict under some circumstances. Many of these products have long life and are upgraded or adapted during their lifetime. They are typically designed by a large number of people often at different locations and distributed over a large supply chain. Unlike much simpler consumer products these products typically only have a small number of competitor products which can be used as guide for the system design.

System architecture design is addressed since the beginning of systems engineering after the Second World War by several communities. The International Council on Systems Engineering (INCOSE)defined a system is a "combination of interacting elements organized one or more stated purposes"(INCOSE 2007). The Institute of Electrical and Electronics Engineers (IEEE) defines system architecture as "the fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolutions" (IEEE 2000). Eppinger and Browning (2012) generalize the IEEE definition further by replacing the word "organization" with the "structure" and the product-oriented terminology of "elements" for any kinds of "components" defining system architecture as "the structure of the system, embodied in its elements, their relationships to each other (and to the system's environment), and the principles guiding its design and evolution – that give rise to its functions and behaviors". This integrates the product view of the system architecture with the design process view of the system architecture. In general, the notion of the "system" can be applied to product, process and organization, or embodiment of the three at the same time.

System architecture design occurs at different levels of detail as shown in Figure 1 The broadest level are so called System of Systems (SoS) concerned with the technical aspects, the procedures and modes of use of large scale human endeavors, such as transport system, air traffic management systems, urban infrastructures, where different technologies and processes need to be get traded off against each other. A System of Systems as defined by Mark W. Maier (Maier 1996) presents five main features: (1) Operational Independence of the Elements: Each system

composing the SoS can operate separately; (2) Managerial Independence of the Elements: Systems have an independent design and acquisition process; (3) Evolutionary Development: The SoS is dynamically formed, systems are added or removed; (4) Emergent Behaviour: The capabilities of the SoS are more than the sum of capabilities of the systems which compose it; (5) Geographic Distribution: The geographic extent means system interactions are based on information and should rely on a network. On a System level high-technology technical systems integrating human component such as aircraft, cars and train (to mention some of the most obvious ones) are often discussed. These technical systems themselves are often composed of multiple very complex subsystems, designed and built by suppliers with multiple customers, which unlike in a SoS can't operate independently, but are still often thought of as systems in their own right. This paper will concentrate of the architecture of systems and subsystems, i.e. predominantly technical systems (see Figure 1), which need to be considered in the context of the SoSs in which they operate.

Figure 1: 1) Different hierarchical levels of system architecture (left side) and 2) Examples of system architectures (right side).

Different communities also have a different focus in their research. Research on Cyber-Physical Systems (CPS) stemming from embedded systems looks at the representation and model heterogeneity in system architecture (Lee 2014). The system engineering community looks at how to manage system architecture and what support is needed both on system level and SoS level. For example, A3 system architecture models allowing sharing information (Brussel and Bonnema 2015). A focus on design synthesis comes from the engineering design community, who are interested in how different representations, methods, and tools can support system architecture decisions and design process. This is a relatively small community, who work closely with industry and therefore address system architecture in a specific context.

In the design literature, the boundaries between conceptual design and system architecture design are not clear nor is the terminology used in companies. One perspective (Baumgarten and Silverman 2007) is to look at system architecture as the overall process from the definition of system boundaries and the selection of fundamental solution principles to the development of the overall design of the product. In which case conceptual design is a stage in system architecture design where different technical configurations are developed. Another perspective is to think of system architecture design being part of conceptual design. One of the reasons for the difference in perspective is that many designs are incremental, so that fundamental decisions about system architecture are inherited from pervious design rather than explicitly addressed by designers.

This paper argues that existing research on system architecture in the design research community has concentrated on computational methods for architecture generation and modes of representation rather than on the variety of system architectures that exist in industry and the specific methods that are needed for particular product or system characteristics. Section 2 explains our methodology. As section 3 illustrates, the majority of academic literature concentrates on generative aspects of system architecture for new system, neglecting incremental design, where partial solutions are carried over. The literature on system architecture, which is discussed in section 4, addresses how system architecture design can be supported through various tools and methods, but is rarely engaged with how the nature of the product and organization affect the system architecture design process. This makes it difficult for practitioners and academic researchers to know when particular tools, methods and insights can be

deployed. Therefore, we discuss in section 5 a classification of types of system architecture based on the characteristics of the products and discuss how system architecture plays out at different levels of detail. Section 6 underlines some of the properties of the different types of system architecture before section 7 discusses the implications for supporting system architecture design process for different categories of system architecture.

2 METHODOLOGY

The literature on system architecture design rarely characterizes the properties of the systems that are being design and therefore is not explicit about the scope the claims being made in the papers. When discussing the findings of their own papers, Wynn et al (2009) and Moullec (2013), the authors released that the processes and the uncertainties that needed to be considered were totally different between the two cases, but familiar to each from other case studies, as diesel engines are an incremental design of a very mature product similar to the automotive design studied in Jankovic (2006). This revealed a clear need to distinguish between different kinds of problems in system design. This prompted a look at the literature for classifications of system architecture problems. As no suitable one was found, the authors decided to build on their own studies, see Table 1. There are at least 10 case studies related to complex system design that are used as a basis for this paper. Most of the studies have been conducted looking at different aspects of complex system design as collaborative decision-making, engineering change, new product or system development processes. The studies have been carried out in collaboration with different companies in different industry sectors using action-research based approaches. These studies let in the majority of cases to methodologies and tools to support different aspects of complex system and architecture design. The aim of this work is to develop a theoretical understanding of design processes based on drivers and constraints on design processes with the view of predicting some of the behavior of design processes from the nature of the product and the starting conditions (Eckert and Stacey 2014).

Table 1: Studies by the authors

Reference	Aspect of system architecture	Industry sectors	Product
Eckert, Clarkson et al. 2004	Incremental design and Engineering change	Aerospace	Helicopters
Jarratt, Eckert et al. 2004	Engineering change	Automotive	Off highway engines
(Jankovic 2006)	Collaborative decision making	Automotive	Passenger cars
Wyatt, Eckert et al. 2009	System architecture design process	Automotive	Off highway engines
Eckert and Clarkson 2010	Process Planning	Automotive	Sports car
Jankovic, Holley et al. 2012; Holley, Jankovic et al. 2014	Constraints in architecture generation	Chemical engineering	Chemical plant

Moullec, Bouissou et al. 2013	Uncertainty and innovation	Defense	Radars
Ye, Jankovic et al. 2014, Ye, Jankovic et al. 2015	Supplier selection	Automotive	Engine
Eckert, Isaksson et al. 2014	Product Platforms	Automotive	Trucks
Ben Hamida, Jankovic et al. 2015	Tools for system architecture	Aerospace	Satellites

The research process that has been followed is shown in Figure 2. Realizing from initial discussion that competing companies had similar concerns and process in defining product architecture, we aimed for a simple distinction between incremental design and designs where solution principles, but few specific components, are reused. However cutting across both categories, some companies put additional effort into assuring that the architectures allow them flexiblity in responding to future needs regardless whether these were incremental designs or those reusing solution principles. Another category was added for product platforms, as commonality across a range of products increased both the uncertainties and the constraints on the product. We also added ab initio design as a category, which is discussed in the literature, even though none of our case studies covered this case. Once we had identified a number of potential factors, we abstracted and structured the factors along the dimensions of time, novelty and integration, as we will explain further in section 4. The classification was refined until we has reached the smallest classification of influencing factors that accounted for our cases and those we had found in the literature. Afterwards, during new projects we have been discussing these factors within a workshop that was conducted on system architecture decision support (Ben Hamida, Jankovic et al. 2015). These discussions suggested refinements and allowed us to partially validate the factors and their impact on system architecture design process.

Figure 2: Research study process

3 APPROACHES TO SUPPORTING SYSTEM ARCHITECTURE DESIGN

Designing complex systems raises particular challenges for designers such as modelling inherent couplings, incorporating uncertainty modelling, large-scale optimization, multi-disciplinary design optimisation and emergent behaviour (Minai, Braha et al. 2006, Bloebaum and McGowan 2010). Distinct and adequate support (Allen, Azarm et al. 2011) is required to link across multiple domains, areas of expertise and use scenarios. This is particularly acute in the design of system architecture where different aspects of a complex system come together for the first time in a design process.

System architecture generation, evaluation and selection is related to a large number of decisions (Fixson 2005) as illustrated in Figure 3 (Fixson (2005)): product related decision like development process and project team; process related decisions concerning design process, development resources, production capacity, assembly

process; and supply chain related decision defining the logistics, suppliers, etc. This is by no means an extensive list and explored further in (Fixson 2005).

Figure 3 : Architecture impact (see(Fixson 2005)

At the beginning of a design process most of these relationships are yet to be explored and defined. Nevertheless, the importance and the impact of decisions related to system architecting are widely acknowledged. Tools to generate system architecture are needed as well as tools, methods and representations that articulate the relationship between the system architecture and other elements of system design process. However, much of the research pertinent to system architecture design, such as functional modelling, has not originally been motivated by system architecture design but was either conducted from a theoretical perspective or to address issues during later stages of the product development process. The majority of the research into support for system architecture design comes from the US tradition of research motivated by theoretical questions, which is illustrated through applications to either illustrative examples or small scale practical problems. In that tradition the problem of the scope of the applicability of the research is rarely posed. Their emphasis is largely on the design of totally new systems, which rarely occurs the systems that we have observed in industry.

The systems architecting process consists of modelling of requirements and constraints, generation of possible architectures and their evaluation with regard to desired performances. Cagan, Campbell et al. (2005) argue that computational design synthesis methods in general need to integrate four main activities: *Representation* of the attributes of the design space (design alternatives, objectives and constraints are specified); *Generation*, which uses this representation to propose candidate solutions; *Evaluation* with regard to final objectives; and feedback from the evaluation called *Guidance*, which is used to steer the search process in subsequent iterations. The element of guidance is often missing in system architecture designs, as there can be a significant delay before the next product generation is started.

An important aspect of system architecture is the able to predict the performance of the product. In this a key element is mapping from the functional domain to the structural domain, and from the structural domain to the behavioural domain. This is supported by several functional modelling approaches developed for slightly different aims. The Function-Behaviour-Structure (FBS) framework (Gero and Kannengiesser, 2004) supports the design of systems with an implied sequence of steps from function to structure to behaviour, which mirrors the steps of system architecture design. The Function-Behavior-State models (Umeda, Ishiia et al. 1996) draws attention to the fact the behaviour of a system can be measured and its state can be observed, but the functions are assigned to a system are to a certain extend personal and subjective. The Structure-Behavior-Function model (Goel, Rugaber et al. 2009) supports functional reasoning. Axiomatic design (Suh, 2001) points out that in practice many design activities involve a rapid move from the structural domain to the functional domain, as the embodiment decisions taken give rise to new functions and behaviour which need to be suppressed or enhanced through further structure. An exhaustive review of functional modelling approaches is beyond the scope of this paper, but has recently been provided by Vermaas (2013) and Crilly (2013).

However, the existence of theoretical models of the mapping between function, structure and behaviour does not translate into practical tools used in industry. There might be several reasons for this; one is that the architecture of a complex system involves hundreds or thousands of parts. If a model represents the relationships linking the elements in two domains, its complexity increases rapidly. There the modelling effort is potentially large. Another reason is that in system architecture, functions are carried out by multiple components and components carry out multiple functions.

Architecture models show at least partial relationships between function, behaviour and structure; and draw on different ways of showing relationships such as graph theory, network theory, matrices or grammar based approaches. Here we discuss different methods used for concept or architecture generation, but this is not an exhaustive literature review. Some research approaches start with mapping from function to structure, while others start with the elements of the product and aim to predict is behaviour.

3.1.1 MAPPING FUNCTION OR TARGET BEHAVIOUR ONTO STRUCTURE

The main idea of mapping function onto structure, is that starting with the product or system objectives or functions, and rules for function structure allocation, it is possible to generate possible structural configurations that satisfy previously defined functions. One of the main motivations for starting with functions (e.g. Pahl and Beitz, 2013 or Suh, 2001) is overcome fixation on existing structures. However in practice expressing product functions can be difficult as there are many different notations of function and modelling frameworks (Vermaas 2013) and designers don't find it easy to apply them consistently (Eckert 2013). The challenge of starting with functions is to assure that the system has the desired behaviour; and most approaches only address mapping function to structure.

Generating design structure from the target behaviour has a long history. Ulrich and Seering (1989) represent the problem as a network of parameters, idealised elements in translational-mechanical, rotational-mechanical, fluid-mechanical, and electrical media. Schematic synthesis generates candidate designs, classifying their behaviours, and then modifying the design in order to satisfy the requirements. Strawbridge, McAdams et al. (2002) used the Functional Basis (FB) theory by Stone and Wood (1999) to develop a concept generator tool to create new concepts based on storing and reusing existing design knowledge. They derive a functional model from the weighted customer needs. Deploying afunction-component matrix concept generation based upon identifying all possible structural configurations related to the defined sub-functions.

Kurtoglu and Campbell (2009) propose an approach based upon the Functional Basis theory (Stone and Wood 1999) which described functional architecture as a set of basic functions linked together by energy, material and signal flows. Kurtoglu and Campbell (2009) supplemented this definition by developing a design repository with 92 component types and their corresponding functions. A set of graph grammar rules is used in this repository to generate a configuration flow graph from a defined functional model. The generated graph represents components linked by predefined flows. Gupta and Okudan (2008) extended this method by proposing a framework that integrates modularization, assembly and variety considerations and yields product concepts with components grouped in modules.

A rule-based repository definitions for function/component allocation is proposed by Bryant, Mcadams et al. (2005) who use a set of DSMs to represent dependencies between functions and components. This allocation allows generation of possible product architectures that satisfy defined functions and functional flows. Helms and Shea (2012) chose the Function-Behaviour-Structure (FBS) model for concept generation (Gero and Kannengiesser 2004). This model has three levels of abstraction correlating to classical design process steps: functional decomposition, allocation of physical effects to functions, and embodiment of physical effects with components. A graph grammar is used to map the initial functional graph into a behavioural graph. The resulting behavioural graph is linked and mapped to structural architecture. The originality of this method lies in the adoption of an object-oriented approach that leads to fewer rules. However, neither of the rule based approaches address issues associated with global system performance.

A methodology for architecture generation for complex systems based on fuzzy logic is used by Bonjour, Deniaud et al. (2009), who base their architecture generation approach developed by Pahl, Beitz et al. (2006). The architecture mapping consists of mapping between requirements and system functions; and afterwards from functions to product components. The authors discuss the deployment of this method in particular for the detailed design phase or the incremental phase where expert knowledge concerning different interactions within the system is available. However, although they take into account functional interaction propagations, they only use mappings from one function to one component. The method clusters the components in different modules to optimize the objective function. Simulation of possible structural architectures is possible to meet different client requirements defined at the beginning of the design. Albarello, Welcomme et al. (2012) proposed an overall approach based on functional requirements, in which an algorithm generates a random set of functional architectures. Based on defined constraints and viability rules, functional architectures are mapped to structural architectures. The approach supports and integrates performance calculations and architecture adequacy relative to defined preferences. An evolutionary algorithm is used in the process to refine and discover more appropriate architectures. This iterative process stops when the proposed architectures achieve sufficient performance. This overall approach evaluates architecture performance but it is noteworthy that components and functions, as well as generation rules, are problem-specific and must be defined before each system generation.

3.1.2 MAPPING STRUCTURE ONTO BEHAVIOUR

As complex systems are rarely developed from scratch, the system architecture design begins with existing system structures into which the innovation is integrated. The analysis of existing systems to identify their existing and potential functions as well as the margins for change that the existing systems have, i.e. the amount by which the current system exceeds is various functional requirements and therefore can absorb a change (Eckert et al. 2004), is an important part of most system architecture problems. Therefore structure is used as the starting point to generate new behaviour.

Wyatt, Wynn et al. (2012) propose an approach to support product architecture generation by a network structure of experts' knowledge of qualitative constraints on the arrangement of components and the interfaces between them. This represents the types of interfaces between components, and applies several types of constraints for architecture generation, that define which component types can be connected via which interface types. The types

of interfaces that are modelled include: structural, behavioural, assignment and geometrical. These interface types are taken into account for product architecture generation but are not integrated with performance estimation.

Many system architecture design system use design structure matrices (DSM) or domain mapping matrices (DMM) (see (Browning 2009) to express the mapping between components and functions.Wyatt, Wynn et al. (2008) capture the rules governing product architectures in a DMM and use a component DSM to compare various architecture concepts. By mapping component alongside component types in parallel with component and interface types, they are able to express constraints governing architecture definition. Although they consider interface types, the approach allows mapping component types and interface types, without requiring design parameter. Sharman and Yassine (2004) propose to integrate three levels in system representation into DSMs: global design rules, interface rules and intramodule design rules. Hellenbrand and Lindemann (2008) use DSMs to support the selection of product concepts. They present a compatibility matrix that captures possible compatibilities and their respective weights in relation to two different product components. In the proposed consistency algorithm, the existence of the interface is taken into account in order to offer all possible product concepts for selection.

Albers, Braun et al. (2011) propose the Contact and Channel Approach (C&C) for system architecture generation for incremental mechatronic systems arguing that the technical system functions need to be considered in a form-dependent way. C&C models represent the interactions between systems, subsystems and parts through working surface pairs (WSPs – geometric interfaces between elements of a physical system or between artefacts and environment) and channel and support structures (CSSs), which are physical components or volumes of liquids, gases or fields directly connecting two WSPs).

Ziv-Av and Reich (2005) develop *the subjective objective system* (SOS) for the generation of optimal product concepts. The SOS approach integrates information on market, organization and technology for generation of product concepts. The authors state that this approach can be extended to accommodate more detailed information on interactions between the components, but the proposed mathematical model would not support such information. Another approach, HSoS method (Rosenstein and Reich 2011), uses genetic algorithms to define the design problem. For each decision variable, a set of genes specifies the ability of the decision variable to satisfy diverse constraints and its contribution to fulfilling product objectives. A genetic algorithm searches for solutions with the best performance by satisfying a weighted objective function, and a Pareto front approach allows the best concept to be identified. However, with large search spaces, simulations must be repeated in order to avoid local optimizations in favour of global optimizations.

Moullec, Bouissou et al. (2013) propose a Bayesian model for system architecture generation starting from system structure. The proposed model integrates uncertainties on interfaces, new technologies and achieved performances. The aim of this approach is to combine data a posteriori from previous projects and data provided by expert estimations related to new technologies. A filtering mechanism is provided that rejects architectures that fail to reach a threshold level of confidence in relation to the requirements.

Kalyanasundaram and Lewis (2014) propose a function based approach for generating the architecture of an integrated product. The idea is to combine products into an integrated product allowing for the exploration of an

architecture which could deliver multiple functions. They propose a function-based approach that evaluates the similarity between two products; and propose a matrix-based approach to generate the overall function structure and map which is mapped to the components of existing products to derive an architecture of the new product.

3.1.3 GAPS IN THE LITERATURE

While most of these approaches use real products as examples they are rarely engaging with the real complexity of industrial practice. Research studies often do not reflect about the particular characteristics of the specific domain that they have addressed and therefore are not clear about how applicable the approach would be in different contexts. However, what are suitable tools for particular problems and when to use them is not clear. Few tools are being deployed in an industrial context, even though industry voices a need for tools and methods. We can see a number of reasons for that:

- The research tools are not designed to handle the complexity of models that companies would require;
- The research tools do not address the mix of existing solutions that are carried over and the novel aspects of solutions that are required in incremental design;
- There are few empirical studies of design practice specifically on system architecture, so that it not clear what aspect of system architecture design companies would like to have support with;
- The research community has not reached a consensus on concepts that are fundamental to system architecture, like the meaning of function.

We believe that there is a need to understand when some tools are appropriate and what still needs to be developed as there is not one system architecture problem or process, but rather several depending on the design context, different product characteristics, position in the design chain and degree of innovation. We will discuss these characteristics in the next section of this paper.

4 CLASSIFICATION OF SYSTEM ARCHITECTURES

On a high level of abstraction, a system architecture design process can be seen as an activity that generates or uses requirements, and translates them into a structural description of the arrangement of components and systems in a product, incorporating an understanding (and sometimes explicit description) of the functions and behaviours that the product will carry out. However, looking at different industrial contexts and design processes reveals that the system architecture design process is not a generic process that can be rolled out in the same way in each organisation and for each product, but depends on the nature of the product and the relationship this product has to other past, present and future products.

This observation has pushed us to try to understand the factors that influence these differences and to try to understand what needs to be taken into account in order to reflect on adopted methods and tools. This classification will look at system architecture from the perspective of where the companies starts, i.e. how much of other products is carried over, as this determines the activities that need to be carried out, their sequence and the information available to them. There are of course also differences in company culture or national ways of working which influence system architecture design. However, as these would typically be shared with other activities in the design process they are considered outside the scope of the paper.

These following characteristics should not be considered as orthogonal in the sense that they are independent and uncorrelated; and the system architecture process is influenced by the mixture of these product characteristics. The relative importance of these factors varies between different products. Moreover, each of these dimensions can be considered as a matter of degree. In this section, we will discuss the observed characteristics of products and propose related product classification of system architectures related to these factors in the next section.

4.1 DIMENSIONS OF THE PRODUCT CLASSIFICATION

While much of the academic research on system architecture, such as (Ziv-Av and Reich 2005, Bonjour, Deniaud et al. 2009, Hellenbrand, Kain et al. 2009, Albers, Braun et al. 2011) is concerned with the generation of completely new systems, this is rarely the case in industry, where most products are to a certain extent reusing components, systems or solution principles. Therefore, the degree of novelty is an obvious dimension of our classification. Reuse was not just reuse over time from previous versions of a system, but also across different products or systems, because components are reused or scheduled to be reused in other products as is the case for platform product, or because the system has to integrate existing solutions in the system architecture planning. Both are forms of integration. A third dimension is the modification that products are subject to over their life cycle. While some products are left unchanged throughout their entire life, such as satellites that are in orbit, others are likely to be modified several times, like aircraft or ships. Coming from a product development and engineering design perspective the viewpoint of our classification is the new product that is designed rather than the entire system of which it is part. There the integration with other products becomes a constraint on the design products rather than a dimension depending on the emphasis.

The following characteristics are properties of the product over its life cycle, as illustrated in Figure 4.

Figure 4: Characteristics of products.

1) Degree of innovation on system level

In practice, very few products represent entirely new designs, designed from scratch. One example of a product with no predecessor is the first space shuttle. However, this design was in part based on aircraft and inherited some of its system architecture from aircraft. Another famous example of an innovative product is the Dyson cylinder vacuum cleaner, which required a new system architecture, as the suction mechanism and dust storage was resolved in a completely different way. However many details, like brushes, were similar to existing products already on the market. Other products are similar to existing products on the market made by competitors, which influence the design of the new product. At the other end of this spectrum are incremental designs where the system architecture largely remains the same. Some of these products are extremely mature products where the fundamental functionality has not changed significantly over generations of products and is therefore known to all participants in the design process.

2) Degree of reuse on component or sub-system level

Some products do not make use of existing components, while others reuse a large percentage of existing components or subsystems. In the first case, the system architecture can be designed freely, whereas in the second case the system architecture inherits constraints from past designs. Often products that are designed this way is because they are too technologically complicated to design otherwise and innovation is concentrated more on a subsystem level or component level, i.e. innovation is concentrated within the architecture of a small number of sub-systems; or because the supplier capabilities do not allow for another design. Sometimes products reuse solution principles rather than actual components or subsystems. This can either be a version of a component or system at a different scale and deployed in a different context; or a matter of designing a new version along known principles but to update specific properties.

3) Degree of integration with other products

Integration of the system architectures of several related products (designed at roughly the same time) is an orthogonal issue to integration within a subsystem. Some products are stand-alone products where the company designs a specific one-off product. However, this design is likely to share at least some components or systems with other products designed by the same company. Other products are based on product platforms, thereby integrating several potential solutions of system architecture. To date, very few companies have achieved complete product platforms, so that some aspects of a system are integrated in product platforms while others are not.

4) Degree of modification over life cycle

Another factor that influences the system architecture process considerably is the life cycle and overall length of the lifecycle of one product. Many products are designed with a particular product architecture and remain like that throughout their entire lifecycle, while others with very long life cycles are modified throughout their life to meet new requirements. In some cases, components or systems are upgraded to bring them in line with new technology. However, particular highly complex products can be subject to major upgrades or refurbishment throughout their lifecycles. For example ships, military equipment (tanks, radars, etc.) and aircraft which have very long life cycles are often refurbished and sometimes repurposed in throughout their life time. This issue is also considered by practitioners as one of the major challenges in addressing the management of overall system architecture.

4.2 PRODUCT CLASSIFICATION

These characteristics affect system architecture development processes, as they determine what the process starts with and what needs to be established as part of the system architecture process; as well as the constraints that need to be considered in the system architecture process. The proposed classification is based upon several studies conducted in different industry sectors in complex system design as well as different studies that have been published in literature pertaining to system architecture design.

The motivation behind this classification is to understand different points of view and drivers with regard to the organisation of this process. Alternative classifications might be equally valid. This classification is seen as a frame for discussion on the scope of insights, tools and methods for system architecture design.

Therefore, the proposed categories are not intended to be mutually exclusive. Rather that the predominant aspect of each system architecture design is discussed. For example, when discussing diesel engines it is considered as an example of incremental design. The basic architecture of diesel engines has changed very little over the last 100 years, with a big leap recently when after-treatment systems had to be included in the product. From the perspective of a diesel engine, the company aims to carry over up to 80 % of the components from the previous generation to reduce development costs and parts in maintenance. However, as they introduce new technology in one engine, they want to use it in engines of a different size, so that they carry the solution principles across. By contrast, jet engines reuse many fewer components from engines of a similar size and performance; the development process is focussed on carrying over solution principles from engines of different sizes. One of the underlying differences is that diesel engine makers have to renew their entire offering frequently as emission regulation changes (see Jarratt et al. 2005), whereas aircraft engines are developed for new generations of airframes when requested by the airframers. The example shows these drivers are a matter of emphasis.

The case studies used as a basis for this study a range of issues (see Table 2).

Table 2: Different case studies and classes of system architecture processes

Case study	Ab Initio design	Incremental Design	Reuse of solution principles	Platforms	Future Flexibility
Aircraft		X			X
Jet engine		X	**X**		
Passenger car		X	X		
Helicopter		**X**			X
Truck		X		X	
Diesel engine		**X**	X		
Sports car			X		

1) Ab initio design: In industrial practice, very few engineering products are designed from scratch even though ab initio design is the focus of many engineering text books (Pahl, Beitz et al. 2006). For example, Pahl, Beitz et al. (2006) describe system architecture generation as part of conceptual design as their second major block of activities after the requirements have been established. They start with the identification of the essential problems, i.e. those parts of the system with no standard solution available. Then they propose developing a solution-neutral function structure before searching for working principles and working structures that can be used to fulfil these functions. In practice, this rarely can be a purely sequential process, as the functions and the way they are described bias the system structure; and the system structure give raise to additional functions or displays behaviours that need to be enhanced or suppressed by supplementary functions. Often the designers do not have sufficient knowledge about subsystems and components and very early testing is needed. In this case, the system architecture design process can be intertwined with the testing process so as to progressively build up knowledge and estimation of overall system performances. Suh (Suh 1990, Suh 2001) therefore speaks of a backwards and forwards process between the function domain and the product domain.

2) Incremental design: In practice, most products however are incremental developments of previous products, where many aspects of the system architecture are deliberately maintained.Wyatt, Eckert et al. (2009) describe the system architecture design process of a diesel engine. Diesel engines have been invented in the 1890s and the fundamental solution principles have remained the same ever since. The general configuration of a diesel engine emerged over a hundred years ago. The fundamental functional architecture and system architecture have remained relatively constant. However, this does not mean that there is no innovation in diesel engines. Major changed have occurred in the 1990s over introducing electric control, and in the last few years with the introduction of new after-treatment systems.

Figure 5 System architecture in incremental design, after Wyatt et al (2009)

Figure 4 shows the process of designing the system architecture of an incremental product. The starting point of the new design is given as well as new requirements, in this case arising from the business, the regulatory environment and the customer. The basic functional decomposition is known though years of experience with products. The case study company only use a functional decomposition in FMEA (Eckert 2013). The basic performance equations of the product are also known and given. The company starts the design of a new generation by using a simple parametric model of an architecture embedding known performance equations and modifying an existing engine to see whether it is possible to meet the fundamental performance requirements. They uses a requirement cascade to establish the basic system architecture using performance equations at different levels of detail while considering the numerous constraints that the new engine needs to meet. As this is an incremental product the company has set stringent targets for the reuse of components. Engineers discuss this in terms of newness, i.e. new components or existing components used in a new ways, e.g. under hotter conditions. For each product generation there is a plan as to which product components will be upgraded and which are reused as constraints in the requirement cascade.

In this case study the designers knew exactly when the system architecture had been completed, but the company did not go through a step in their design process that they described as a system architecture phase. The mapping between functions and components was implicit in meeting the newness constraints. This does not mean that the company does not take explicit system architecture decisions. As diesel engines are very tightly packaged, they have to make many decisions about configuration when they run into problems, for example if components have increased in size and therefore block access to key components for maintenance. As current diesel engines require large after-treatment filters, which exceed the envelope of the current product, the engines are now integrated in the products they power to a much larger extent. Therefore, the boundary of the system architecture has changed and the company involves its customers in a previously unprecedented way. The design of the after-treatment filters themselves has been an ab initio design problem for diesel engine companies which they had to resolve in close conjunction with their suppliers.

In summary, incremental designs have a largely known system architecture and therefore do not require an explicit functional structure. The system architecture design process is benefits from very mature parametric models of

system architecture. The products are already highly optimized, and need to meet tight constraints." Considerable effort might be required to integrate innovative features into the existing system architecture.

3) Reuse of the same solution principles: Another form of reuse is the reuse of the predominant solution principles rather than of specific components or sub-systems. In this case, the system architecture and manufacturing processes might be very similar, but the component themselves are different. Hence the new design is not constrained in the same way as one that reuses existing components around which existing ones have to be designed. Jet engines are an example of the reuse of solution principles. Companies like Rolls Royce offer a range of engines for different application and different aircraft types, which they build over long periods during the life cycle of the airframe (see (Kerley, Wynn et al. 2011), for a description of our study of conceptual design of jet engines). When the aircraft makers bring out new aircraft, the engine builders develop new engines. As the aircraft are replaced slowly, it might be a long time since they last designed an engine of that particular size and profile, so that technology has moved on and they cannot simply modify the previous design with a similar specification. Therefore, they start by scaling the design of their most recent engine up or down thereby incorporating the new technology. As engine developers are also under huge pressure to reduce emissions and fuel consumption, each new engine tends to push the envelope of what is technologically feasible. Engines by and large keep the same functionality over generations, but might add new features, for example specific additional energy take off for the airframe. Therefore, the engine manufacturers do not generate a functional model of the engine and usually carry the basic configuration across with slight variations. The function structure mapping might already be done and can be reused with slight variations; or in other cases the designers start directly with the structure. For many subsystems they have a number of solution principles or design variations they could use. Many of the negotiations during the system architecture process is around the combination of these possibilities in a new design.

As jet engines are very different in size, sharing physical components with smaller or larger engines is difficult and components of the right size usually belong to an outdated product. However, learning from recent engines is a critical part of designing the next generation. In particular, materials are used across many different engines. Engines are serviced very regularly and components and sub-systems are replaced before they are worn out. Maintainability in terms of monitoring and assessing the condition of the engine, access for inspection and ease of component replacement is a major aspect of jet engine design. As the service process is a large part of the company's revenue, designing the service offering is an integral part of the system architecture design; and trade-off decisions between hardware and service processes need to be considered, for example whether components are integrated in a way that they are replaced together. Of course the engine builders also try to achieve commonality amongst their engines in components and software. However, they would not compromise the performance of the engine to increase commonality, as the automotive industry does.

In summary, products that reuse solution principles on a high level start the system architecture process with a good understanding of the functional structure and the basic configuration of the product. They are likely to draw on several existing products for solution principles and existing technologies and introduce new technologies into several new products. One of the challenges lies in integrating different parts of the systems and assessing the effect they have on each other as well how changes would propagate between different parts.

4) Platform products: In the race for cost reduction as well as diversification of product offerings, many companies producing complex systems have created product platforms. The automotive industry is very advanced in this area, which have specific platforms of for particular types of cars and reuse many components across all of their cars. Each platform sets the target for components to be reused and therefore not necessarily needing design.

There is an inherent contradiction between optimising an individual product, where all components would meet the requirements but not exceed them, and platform optimization, which optimizes the degree of commonality (Isaksson, Lindroth et al. 2014). Most platforms are developed over a period of time starting with a common application and working towards less frequently sold options. Every time an option is added the compatibility between this option and existing options needs to be checked, so that no impossible products are accidentally sold.

Planning the system architecture of platform products is thus an essential part of designing the platform itself. If a product platform is designed from scratch, the company needs to have a clear picture of the market requirements and its own strategy in the market. It also needs to understand the prices it can command for products in the market, so that it can plan where it will make a profit and what products and services might lead to a loss if looked at isolation. Knowing the market, cost and use profile, the company needs to make a strategic decision about which components or systems will be identical in all applications to minimize costs and which will be offered in a range of different applications. For example, a truck company uses a single version of some components, like mufflers, for all applications, and offers a range of options for systems like engines that can be used in all applications. For other components, it offers various options which might not be usable in all applications, e.g. it has several different suspension systems. Other components are only applicable to particular type of applications; e.g. offering options for a lifting truck. This places a particular onus on the design of interfaces between systems and between individual components, which need to be clearly defined and strictly adhered to.

Few companies have the luxury to design a product platform from scratch. In most cases platforms evolve between product generations with some components being carried over from a past platform. These then become constraints on the design of the new platform that need to be accommodated. The other challenge is to introduce product platforms into product offerings that do not use platforms. The companies need to target components or systems they would like to have in their platforms and then design them in a way so they can be used in a number of different applications. This can create linkages that have not existed before not only between different products but also between their design processes. Often this requires the system architects to rethink the system boundaries and use conditions of components, subsystems and products.. The rest of the product is then designed around components that are given, and they become constraints in the design process.

In summary, product platform design requires a large degree of strategic planning across a variety of products and a clear definition of the interfaces between different components, and compromises cannot be negotiated locally. In system architecture design, there is a need for methods that will support the management and cascading of constraints and the simulation of the propagation of the impact of design changes, while supporting the integration of diversity in configurations.

5) Future flexibility: Some high-tech products or systems, in particular military products such as tanks or satellites, have long expected lives and will need to meet future requirements that are difficult to anticipate, so

that the product needs to be designed to be flexible. Their planned use scenario and potential change of use is so diversified that system architects need to imagine possible future changes in order to integrate them into the initial system architecture. Other examples include systems like oil platforms or power plants, which will have to be modified for different applications. In some cases future changes can be anticipated for example the planned introduction of the next stage of an infrastructure project. Design decisions can be taken during the initial system architecture to accommodate future changes either in the sense of design options (DeNeufville, de Weck et al. 2004) where product margins are planned into the system at the beginning, or modular designs where modules can be swapped or the interfaces to planned or potential additions are put into place.

These types of products are often only loosely based on previous designs and are often one-off or low volume products, so that requirements and compromises can be negotiated with a small group of known stakeholders during the system architecture process. The system architecture design process is impacted by the degree of variability and specific methods and tools are required. There are some methods that support understanding this variability and flexibility like (DeNeufville, de Weck et al. 2004, Cardin 2013). However, when it comes to large high-tech projects there are several sources of several sources of uncertainty and reasons to need flexibility; and methods and tools supporting system architecture need to model uncertainties explicitly and assess potential solutions against these uncertainties.

5 INTERCONNECTED SYSTEM ARCHITECTURES

It is often not enough to consider system architecture for a single product or product family. Its sub-systems can also be seen as products in their own right that go through a system architecture process. For a complex product the supply chain also be seen as a system that needs architecting.

5.1 ARCHITECTURE ON DIFFERENT LEVELS OF HIERARCHY

The previous section was discussing system architecture from the perspective of the overall product. System architectures consist of several levels, as illustrated in Figure 1. Therefore, even on the level of subsystems, there is the notion of system architecture and the design process that is related to it. In practice the design process of a complex system is linked to several system architecture design processes depending on the level of the overall system architecture considered. For example the Original Equipment Manufacturer (OEM), for example the airframer, defines the overall system architecture and defines requirements and constraints, that are then cascaded onto a subsystem level, for example an aircraft engine. However, the aircraft engine is designed by independent companies, who have their own system architecture design processes. The aircraft is also constituted of several other system architectures, such as the cockpit or the control system, that are developed in different companies.

The different subsystems fall into different categories according to the classification introduced in the previous section. For example a manufacturer of construction equipment designs the system architecture for a digger. The construction equipment company uses a product platform for many of its components; the cockpit is likely to be a platform component and is shared across many different kinds of construction equipment. The engine is typically a standard engine either designed in-house as a platform component or bought in from an engine manufacturer, and is designed incrementally from the previous generation of engines. The fuel pumps in the engine might be

standard components that are shared across many applications including automotive and completely optimised for mass production.

The degree of innovation and the level of ab initio design also varies significantly between products. Products that are seen as an incremental designs often have highly innovative components or subsystems; and overall innovative products can be made from fairly standard elements. This points to a highly uneven level of risk associated with different parts of the product as well as an uneven quality of information available to the system architect, so that the steps of the design process and the decisions that need to be made can vary between parts of the products. It is therefore difficult to speak of a single system architecture process for a company or a product type, as it varies with the uncertainty associated with a component and degree of maturity that it has. Many system architecture decisions are relational (compatibility between components, change propagation etc.). In some case these relations are known when the product architecture is designed; in others they can be controlled through the design decisions that are made later on.

System architecture decisions propagate across the different levels of hierarchy. The top level decisions trickle down to components on a detailed level. However detailed issues can also propagate upwards. For example, when components are reused their constraints and performances also determine the interfaces that define the design at a top level; hence fundamental design options are cut out by details that the company is already committed to. What is inherited and what innovation is integrated will influence the order in which system architecture decisions need to be taken. In industry, most of the time there is a notion of a certain sequence of decisions that is standard. The difficulty is that this sequence is not necessarily adapted to the project and a given system architecture, which can cause considerable time delays.

Another inherent difficulty related to the decision making process in system architecture design is that different teams are working on different system architectures at the different levels of the overall system architecture. Distributed teams and communication issues hinder the overall system architecture design process. We believe that tools and procedures are needed for sharing these constraints across different system architectures and investigating impacts across different levels. Methods that will also support reflection on the system architecture decision order are essential for the design process.

5.2 DESIGN SUPPLY CHAIN

In very large systems, the design is often done either by the OEM context or in close conjunction with key suppliers in what is also called extended enterprise context. Therefore, the design of one system architecture is often distributed across this design supply chain. The configuration of this chain influences system architecture as it defines system boundaries. There is typically a time delay between the system design of the customer and that of the supplier, as the company needs to know what they want from their supplier. This configuration also defines the propagation of requirements and constraints, thus also influencing the way one designs system architecture. To a certain extent all suppliers receive requirements from their customers, but in practice they often need to work with customers to establish the requirements that they have. For long lead time items orders for components or systems have to be placed before the system architecture process is finalized, so that interfaces to

the component or sub-system are frozen and the rest of the product is designed around it (Eger, Eckert et al. 2005).

Suppliers and customers are often integrated closely in the system architecture process. For example, the diesel engine company has to work closely with a variety of customers to accommodate the rather large after-treatment systems in the existing product architecture of their customers or work with the customers to optimize their architectures to accommodate the larger engines. On the other hand, suppliers and customers are often treated as "black boxes" where interfaces are defined in the system architecture process. Therefore, the design of one system architecture considers the boundaries and interfaces that are defined without providing all necessary information on system performances or on constraints that are propagated. A "black-box" approach does not allow trade-offs on different levels of system architecture hierarchy. In very complex products, this can be an effective way of reducing the complexity a company needs to handle, but it also means that nobody has an overview of the product at any meaningful level of detail, which increases the burden on validation later in the design process.

6 DISCUSSION

The academic literature described in section 3 describes system architecture design as the process by which the structure of the product is derived from the functions that it needs to fulfil, as shown in the left hand side of **Erreur ! Source du renvoi introuvable.** In practice however, functional models are rarely used (Eckert 2013) and companies start from lists of requirements describing target behaviour, performances or properties. Requirements are often linked directly to an existing structure, with a known behaviour, which gives the company an understanding of achievable future behaviour. The perfect example is the design of diesel engine, where the design process begins with a requirement cascade from which modifications to a partially pre-existing structure arises. However, the objective of a design processes is not implementing the functions but reaching the desired behaviour and thereby satisfying the given requirements. Therefore, engineers look to map key design parameters in parametric models at the system level (see left side of the Figure 6), relying on expert knowledge without an explicit model of the mappings between the structure and these key parameters. Mapping these relationships is difficult as some of the information is only available later in the design process.

Figure 6: Definition of system architecture in academic literature (left) and in industry (right)

In industry, system architecture design is embedded in the broader activities of one organisation. A project management team works in parallel to the system architecture design activities. They deal with the strategic planning in the organisation, allocate resources and manage the costs. They provide constraints and targets for system architecture design and need to assure that the implementation of the system architecture can take place from a commercial and practical perspective. In practice in different companies the system architecture activities run in parallel or overlap with other design activities. Designers understand the existing products and technologies and can advise system architects on what is technically feasible on a system, subsystem and component level. They provide constraints to the system architecture process, but also highlight possibilities where architectural changes can be carried out without significant effort or the need to introduce risky innovations.

The differences between products in the different classes depend on the difference between the baseline structure and the target structure, as illustrated in Figure 7 :

- Ab initio designs: these are very rare. Here the starting point is not a base line design, but the designers go from function to structure.
- Incremental design starts with an analysis of the baseline design and aims to maximize the reuse of elements, so that the baseline structure and the new structure will share many elements.
- Reuse of solution principles also starts with an analysis of a baseline design, but rather than carrying components and systems across only solution principles are carried over.
- Product platform products start with an assessment of function requirements and the existing platform's ability to meet before new components are designed. These elements are used to build up a structure. Note that many platform products also have a similar product architecture, so that there is a baseline.
- Future flexibility products are likely to go through one or more cycles of system architecture revision throughout the product life cycle.

Figure 7 System architecture in the different classifications

Figure 8 shows project management, system architecture design and design as parallel activities. In some organizations, such as aircraft builders or large automotive companies, they are carried out by different teams with distinct sets of expertise: project management is carried out by business people or a team of engineers; system architecture by a systems team consisting of system architects and domain experts, who have deep knowledge of particular aspects of a design; and design being carried out by a broader team of designers from particular disciplines.

However, in smaller companies system architecture is carried out by a small group of designers with inputs from many of their colleagues.

Figure 8: Relationship between System architecture, Product design and Project management

In most of the cases, the system architecture is incremental. Most tool and methods however start with generated solutions and go through a function, structure and behaviour iteration. However, in industry, system architecture teams need to consider two system architectures at once: baseline architecture, i.e. the starting design, and the new architectures. This is in part due to differences being managed through new requirements. However support for mapping the difference between the baseline and new architecture is missing in methodologies and support tools (see dashed line in Figure 6).

The proposed product classification of system architecture design processes is not necessarily exhaustive. It cuts across several industries and domains that have been observed. We believe that this classification needs to be discussed and challenged to identifying further factors hat impact system architecture design. However a classification is necessary, because of its obvious impact on system architecture design. **The order of the**

decisions that need to be made at the system architecture level is different in different product classes, therefore which methods and tools are relevant and when they are used is not the same. We believe that there is a need to discuss this issue with regard to **decision support tools** used for system architecture design.

The elements of system architecture are in most of the cases the same and some can be considered as generic (functions, components, system parameters, etc.) However, there are differences in inherited structure, the level and the position of the subsystem architecture within the overall system architecture, the economic and business factors that influence the design supply chain as well as the system architecture itself, etc. There is a need for a comprehensive framework that will recognise variability in the design processes of system architecture and support the design teams in adapting and using appropriate methods and tools. In relation to system architecture design processes and product classifications, different product characteristics will have an impact on:

- **Data:** The quality and quantity of data needed for system architecture design will be different. In some cases a considerable amount of data exits and the requirements cascade is known through which the consequences of the data should be propagated; therefore methods and tools that will capture and use this data will be necessary. In other cases, the quality of data can be variable hence methods and tools supporting different precision of data and mixing quantitative and qualitative data will be needed.
- **Activities that need to be carried out:** the activities related to the system architecture design can be different for different types of product. For example, if a functional description of the system architecture already exists, methods that will support architecture generation starting from function are needed. However, system architecture can start from structure or even from operating conditions, requirements or customer services. These can operate as proxies for missing functional information. This requires different strategies in terms of the activities required and the order in which they will be performed.
- **Tools:** Different product characteristics will influence considerably the type of tools that are needed at the system architecture level. For very large and distributed system architectures, visualisation tools that provide an overview and allow information sharing are essential. For more incremental design, tools that are used in Product Life Cycle Management could be used and enhanced.
- **Order of decisions in the system architecture design process:** The order in which decisions are taken is influenced by inherited constraints, the place of interfaces and their definition, the propagation of requirements, and necessary innovation. This order for one type of product can be known from previous experience, but even then needs to be adapted with regard to product characteristics (see section 4.1). The fact that some decisions commit resources and cannot be revoked is inescapable in system architecture design. The propagation of impacts of decisions on system architecture needs to be taken into account. Therefore the order in which decisions are made should also be considered in system architecture design processes. To our knowledge, no methods and tools have considered this issue previously.

7 CONCLUSIONS

System architecture design is an essential part of any of complex system design processes, however how it is played out in detail varies between processes. This paper has discussed different categories of system architecture design processes based on several research studies in different companies and industrial contexts. The starting point for system architecture design processes varies depending on the degree of innovation in the product, ranging from (rare) complete new designs to incremental designs with targeted innovation; and on the degree of interaction with other products that are developed at the same time. System architecture design is also carried out at different scales from whole complex product to systems within the product.

As a community, we require a better understanding of the differences between system architecture processes and the needs of the engineers involved in them to be able to provide them with better tools or methods. As many of the fundamental decisions that involve both properties of the product and the process by which it is created are set during system architecture design, a better understanding of system architecture design is a key to better design across the product development process and the product life cycle.

8 REFERENCES

Albarello, N., J.-B. Welcomme and C. Reyterou (2012). A formal design synthesis and optimization method for systems architectures. *9th International Conference of Modelung, Optimization and Simulation MOSIM'12. Bordeaux*, France.

Albers, A., A. Braun, E. Sadowski, D. Wynn, D. Wyatt and J. Clarkson (2011). "System Architecture Modeling in a Software Tool Based on a Contact and Channel Approach (C&C-A)." J*ournal of Mechanical Design **133**(10): 101006 101001-101008*.

Albers, A., A. Braun, E. Sadowski, D. Wynn, D. Wyatt and J. Clarkson (2011). "System Architecture Modeling in a Software Tool Based on a Contact and Channel Approach (C&C-A)." *Journal of Mechanical Design **133**(10).*

Allen, J., S. Azarm and T. Simpson (2011). "Designing Complex Engineered Systems." *Journal of Mechanical Design **133**(10): 100301-100301*.

Baumgarten, E. and S. J. Silverman (2007). Dynamic DoDAF and Executable Architectures. *Military Communications Conference, 2007*. MILCOM 2007. IEEE.

Ben Hamida, S., M. Jankovic, M. Callot, A. Monceaux and C. Eckert (2015). Towards a System Architecture Design Decision Support Framework. *International Conference on Engineering Design, Milan, Italy.*

Bloebaum, C. L. and A.-M. R. McGowan (2010). "Design of Complex Engineered Systems." *Journal of Mechanical Design **132**(12): 120301-120301*.

Bonjour, E., S. Deniaud, M. Dulmet and G. Harmel (2009). "A Fuzzy Method for Propagating Functional Architecture Constraints to Physical Architecture." *Journal of Mechanical Design **131**(6): 061002-061002.*

Browning, T. R. (2009). "Applying the design structure matrix to system decomposition and integration problems: a review and new directions." *Engineering Management, IEEE Transactions on **48**(3): 292-306.*

Brussel, F. F. and G. M. Bonnema (2015). "Interactive A3 Architecture Overviews: Intuitive Functionalities for Effective Communication." *Procedia Computer Science **44**(0): 204-213.*

Bryant, C., D. A. Mcadams and R. B. Stone (2005). A computational technique for concept generation. *ASME International Design Engineering Technical Conference & Computers and Information in Engineering Conference* Long Beach, USA.

Cagan, J., M. I. Campbell, S. Finger and T. Tomiyama (2005). "A Framework for Computational Design Synthesis: Model and Applications." *Journal of Computing and Information Science in Engineering **5**(3): 171-181*.

Cardin, M.-A. (2013). "Enabling Flexibility in Engineering Systems: A Taxonomy of Procedures and a Design Framework." *Journal of Mechanical Design* ***136****(1): 011005-011005.*

Chepko, A., O. De Weck, W. Crossley and D. Linne (2008). A Modeling Framework for Applying Discrete Optimization to System Architecture Selection and Application to In-Situ Resource Utilization. *12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference,* Victoria, British Columbia Canada.

Crawley, E. F. (2007). ESD.34 Systems Architecting—Lecture Notes. IAP, MIT Engineering Systems Division.

Crilly, N. (2013). "Function propagation through nested systems." D*esign Studies* ***34****(2): 216-242*.

DeNeufville, R., O. de Weck, D. Frey, D. Hastings, R. Larson, D. Simchi-Levi, K. Oye, A. Weigel and R. Welsch (2004). Uncertainty management for engineering systems planning and design, Massachussets Institute of Technology.

Eckert, C. (2013). "That which is not form: the practical challenges in using functional concepts in design." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* ***27****(3).*

Eckert, C. M. and M. K. Stacey (2014). Constraints and Conditions: Drivers for Design Processes. *An Anthology of Theories and Models of Design. A. Chakrabarti and L. T. M. Blessing,* Springer London**:** 395-415.

Eger, T., C. Eckert and J. P. Clarkson (2005). The role of design freeze in product development. *15th International Conference on Engineering Design,* Melbourne, Australia,.

Eppinger, S. D. and T. R. Browning (2012). Design Structure Matrix Methods and Applications. Cambridge, MIT Press.

Fixson, S. K. (2005). "Product architecture assessment: a tool to link product, process, and supply chain design decisions." *Journal of Operations Management* ***23****(3–4): 345-369*.

Gero, J. S. and U. Kannengiesser (2004). "The situated function–behaviour–structure framework." *Design Studies* ***25****(4): 373-391*.

Goel, A. K., S. Rugaber and S. Vattam (2009). "Structure, behavior, and function of complex systems: The structure, behavior, and function modeling language." *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* ***23****(Special Issue 01): 23-35.*

Gupta, S. and G. E. Okudan (2008). "Computer-aided generation of modularised conceptual designs with assembly and variety considerations." *Journal of Engineering Design* ***19****(6): 533-551*.

Hellenbrand, D., A. Kain and U. Lindemann (2009). Systematic identification of representative solutions to support the concept selection phase *International Conference on Engineering Design, ICED 09*. Stanford, CA, USA.

Hellenbrand, D. and U. Lindemann (2008). Using the DSM to Support the Selection of Product Concepts. *10th International Design Structure Matrix Conference*, Stockholm, Sweden.

Helms, B. and K. Shea (2012). "Computational Synthesis of Product Architectures Based on Object-Oriented Graph Grammars." J*ournal of Mechanical Design* ***134****(2): 021008-021008*.

IEEE (2000). Recommended Practice For Architectural Description of Software-Intensive Systems, *Instutute of Electrical and Electronics Engineers Strandards Association (IEEE).*

INCOSE (2007). Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, *International Council on Systems Engineering (INCOSE).*

Isaksson, O., P. Lindroth and C. M. Eckert (2014). OPTIMISATION OF PRODUCTS VERSUS OPTIMISATION OF PRODUCT PLATFORMS: AN ENGINEERING CHANGE MARGIN PERSPECTIVE. DESIGN 2014 *13th International Design Conference*, Cavtat, Croatia.

Jankovic, M. (2006). Collaborative decision making in new product development. Application to the car industry. PhD Thesis, Ecole Centrale Paris.

Kalyanasundaram, V. and K. Lewis (2014). "A function based approach for product integration." *Journal of Mechanical Design* ***136****(4): 041002*.

Kerley, W., D. C. Wynn, C. Eckert and J. Clarkson (2011). "Redesigning the design process through interactive simulation: a case study of life-cycle engineering in jet engine conceptual design." *International Journal of Services and Operations Management* ***10****(1): 30-51*.

Kurtoglu, T. and M. I. Campbell (2009). "Automated synthesis of electromechanical design configurations from empirical analysis of function to form mapping." *Journal of Engineering Design* ***20****(1): 83-104.*

Lee, E. A. (2014). Constructive Models of Discrete and Continuous Physical Phenomena.

Maier, M. W. (1996). "Architecting Principles for Systems-of-Systems." *INCOSE International Symposium* ***6****(1): 565-573*.

Minai, A. A., D. Braha and Y. Bar-Yam (2006). Complex engineered systems: A new paradigm. *Complex engineered systems,* Springer**:** 1-21.

Moullec, M.-L., M. Bouissou, M. Jankovic, J.-C. Bocquet, F. Réquillard, O. Maas and O. Forgeot (2013). "Toward System Architecture Generation and Performances Assessment Under Uncertainty Using Bayesian Networks." *Journal of Mechanical Design* ***135****(4): 041002-041001*.

Pahl, G., W. Beitz, J. Feldhusen and K.-H. Grote (2006). *Engineering design: a systematic approach.* London, UK, Springer-Verlag.

Sharman, D. M. and A. A. Yassine (2004). "Characterizing complex product architectures." *Systems Engineering* ***7****(1): 35-60.*

Stone, R. B. and K. L. Wood (1999). "Development of a Functional Basis for Design." J*ournal of Mechanical Design* ***122****(4): 359-370.*

Strawbridge, Z., D. A. McAdams and R. B. Stone (2002). A computational approach to conceptual design. *ASME 2002 Design Engineering technical Conference,* Montreal, Canada.

Suh, N. (1990). *The Principles of Design*, Oxford University Press.

Suh, N. (2001). *Axiomatic Design: Advances and Applications.* Oxford, Oxford University Press.

Ulrich, K. (1995). "The role of product architecture in the manufacturing firm." *Research Policy* ***24****(3): 419-440.*

Ulrich, K. and W. Seering (1989). "Synthesis of schematic descriptions in mechanical design." *Research in Engineering Design* ***1****(1): 3-18.*

Ulrich, K. T. and S. D. Eppinger (1995). *Product Design and Development.* New York, Irwin McGraw-Hill.

Umeda, Y., M. Ishiia, M. Yoshioka, Y. Shimomura and T. Tomiyama (1996). "Supporting conceptual design based on the Function-Behavior-State modeller." A*rtificial Intelligence for Engineering Design, Analysis and Manufacturing* ***10****(04): 275-288.*

Vermaas, P. E. (2013). "The coexistence of engineering meanings of function: Four responses and their methodological implications." A*rtificial Intelligence for Engineering Design, Analysis and Manufacturing* **27***(Special Issue 03): 191-202.*

Vermaas, P. E. (2013). "Functional descriptions in Engineering." A*rtificial Intelligence for Engineering Design, Analysis and Manufacturing* ***27****(3).*

Wyatt, D., D. Wynn and J. Clarkson (2008). Synthseis of Product Architecture using a DSM/DMM Based Approach. *10th International Design Structure Matrix Conference,* Stockholm, Sweden.

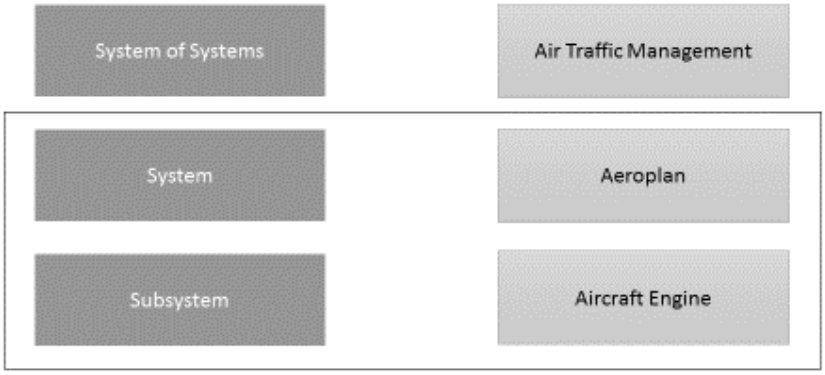
Wyatt, D., D. Wynn, J. Jarrett and P. Clarkson (2012). "Supporting product architecture design using computational design synthesis with network structure constraints." *Research in Engineering Design* ***23****(1): 17-52.*

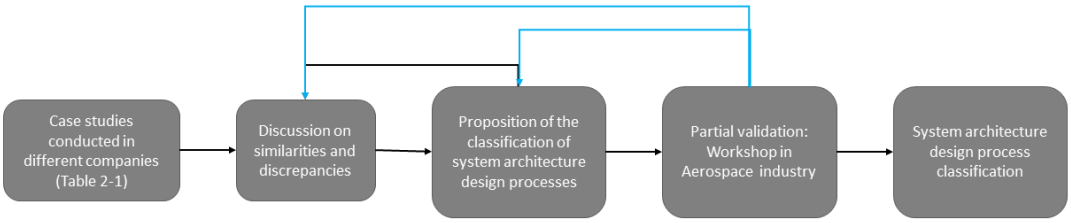
Wyatt, D. F., C. M. Eckert and P. J. Clarkson (2009). Design of product architectures in incrementally developed complex products. *17th International Conference on Engineering Design' ICED'09,* Stanford, California, U.S.A.

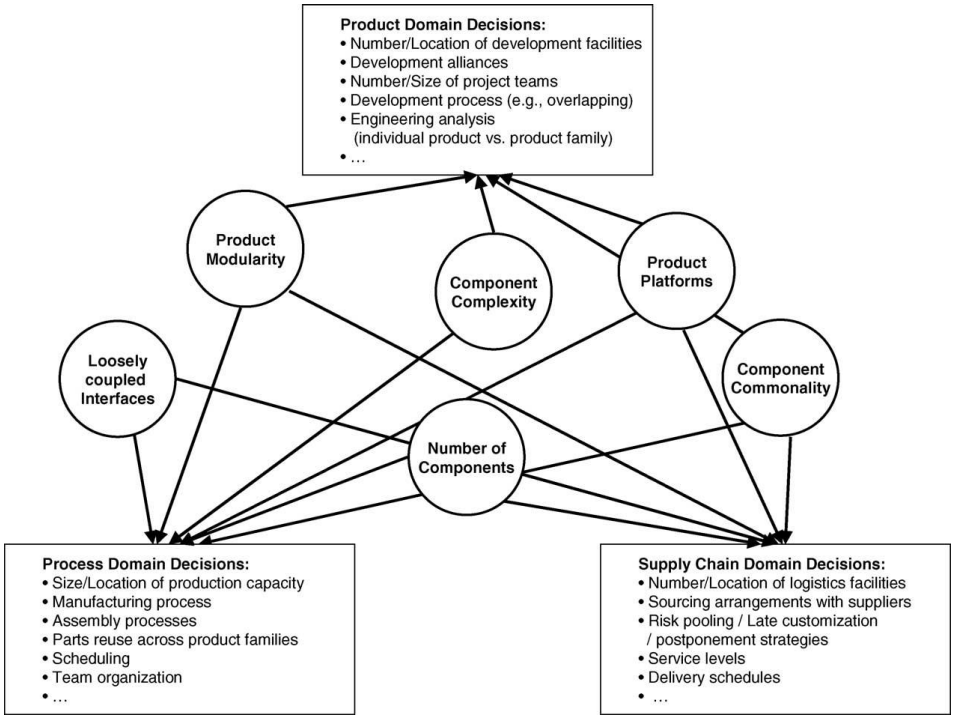
Ziv-Av, A. and Y. Reich (2005). "SOS - subjective objective system for generating optimal product concepts." *Design Studies* ***26****(5): 509-533.*

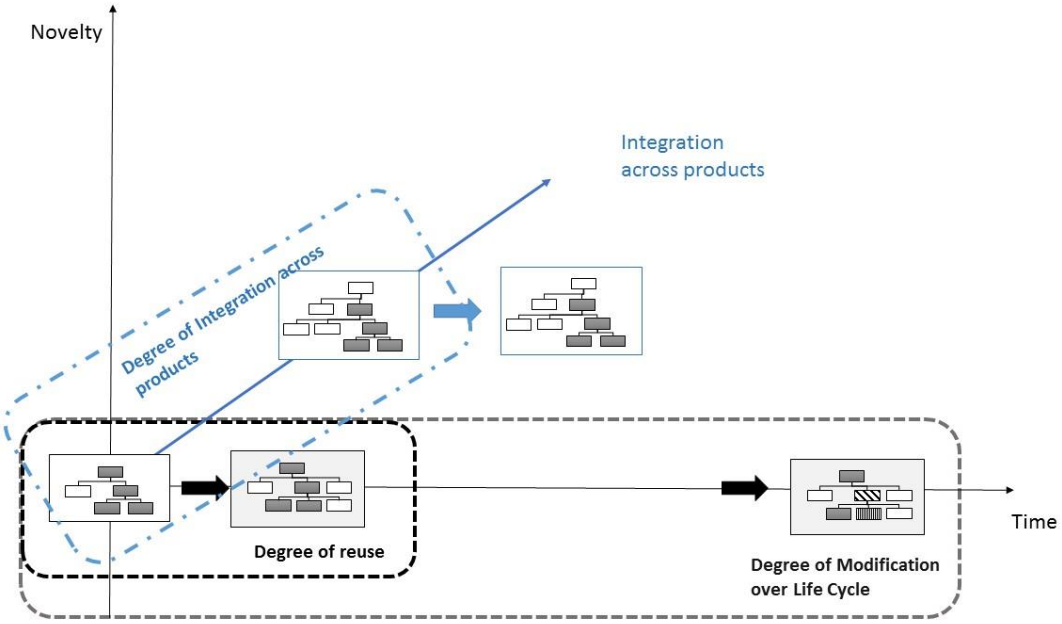
Dr. Marija Jankovic is an Associate Professor at CentraleSupélec, Université de Paris Saclay Her main domain of interest concern developing a decision support framework for early design stages. She is interested in developing support methods and tools that will permit to design engineers to make more robust decisions. Moreover, the research work is also challenged by multi disciplinary design environment that are particularly developing in view to the new world's competition. Dr. Jankovic has a working experience in designing complex systems. In majority of cases, the research projects are done in collaboration with industry or government; and with direct implementation and verification of research results. Therefore, Dr. Jankovic collaborates with some of the major French and international companies: Snecma, Thales, EADS, PSA Peugeot Citroen, Schlumberger, etc.

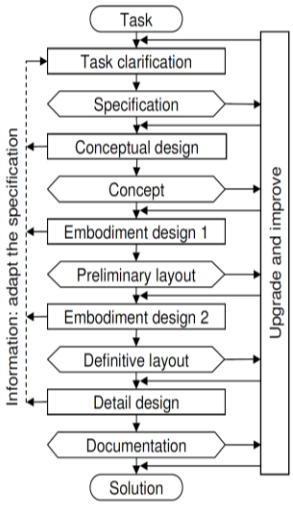
Claudia Eckert is Professor of Design at The Open University, the UK distance education university. She has a long standing interesting in studying and support industrial practice in different design domains and published numerous papers on it. In particular she has been working on process modeling, engineering change and functional modeling of complex engineering products.

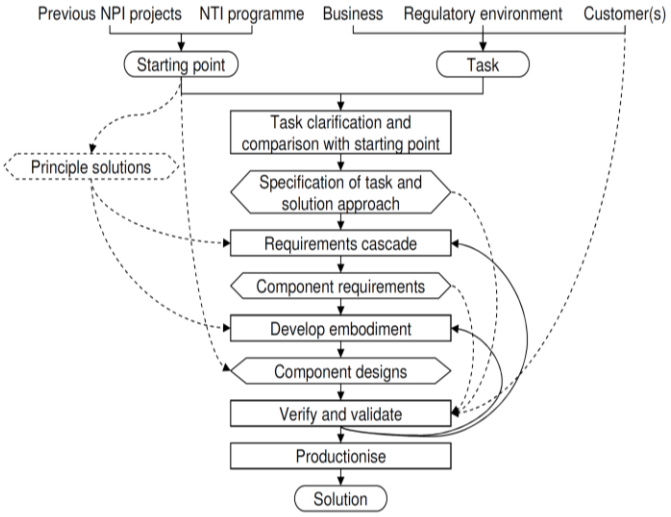

System of Systems
Air Traffic Management
System
Aeroplan
Subsystem
Aircraft Engine

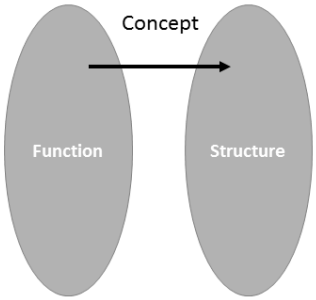

Case studies conducted in different companies (Table 2-1)
Discussion on similarities and discrepancies
Proposition of the classification of system architecture design processes
Partial validation: Workshop in Aerospace industry
System architecture design process classification

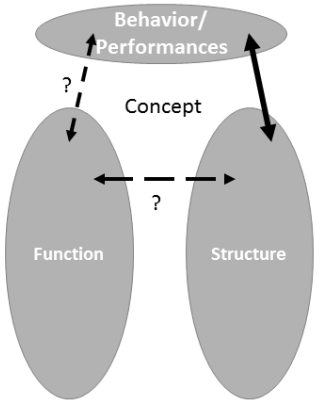

Product Domain Decisions:
• Number/Location of development facilities
• Development alliances
• Number/Size of project teams
• Development process (e.g., overlapping)
• Engineering analysis
(individual product vs. product family)
• …
Product Modularity
Component Complexity
Product Platforms
Loosely coupled Interfaces
Component Commonality
Number of Components
Process Domain Decisions:
• Size/Location of production capacity
• Manufacturing process
• Assembly processes
• Parts reuse across product families
• Scheduling
• Team organization
• …
Supply Chain Domain Decisions:
• Number/Location of logistics facilities
• Sourcing arrangements with suppliers
• Risk pooling / Late customization
/ postponement strategies
• Service levels
• Delivery schedules
• …

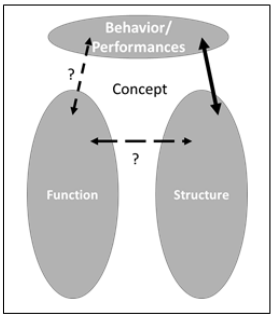

Novelty
Integration
across products
Degree of Integration across
products
Degree of reuse
Degree of Modification
over Life Cycle
Time


Task
Task clarification
Specification
Conceptual design
Concept
Embodiment design 1
Preliminary layout
Embodiment design 2
Definitive layout
Detail design
Documentation
Solution
Information: adapt the specification
Upgrade and improve

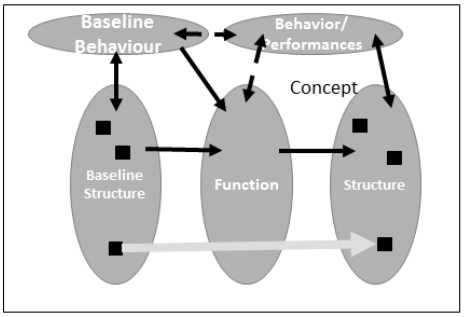

Previous NPI projects
NTI programme
Business
Regulatory environment
Customer(s)
Starting point
Task
Task clarification and comparison with starting point
Principle solutions
Specification of task and solution approach
Requirements cascade
Component requirements
Develop embodiment
Component designs
Verify and validate
Productionise
Solution


Concept
Function
Structure

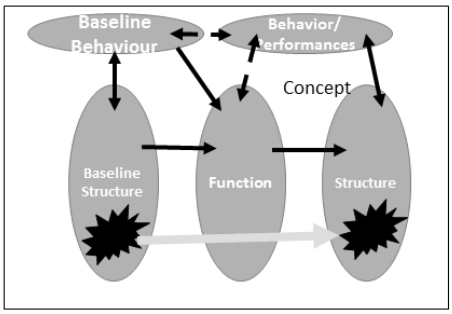

Behavior/
Performances
?
Concept
?
Function
Structure



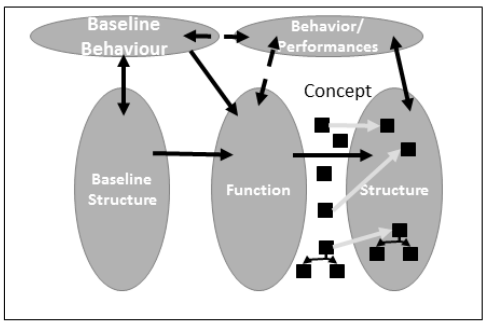
ABINITION DESIGN



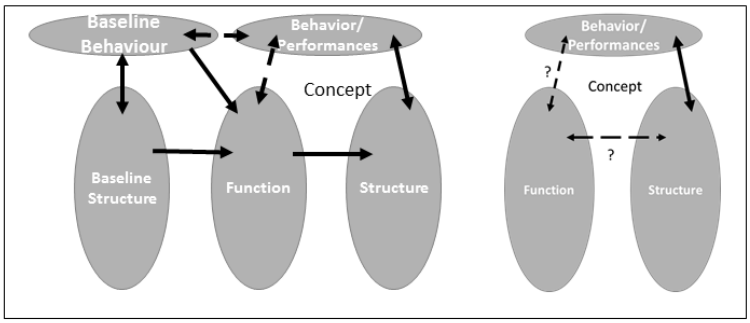
INCREMENTAL DESIGN



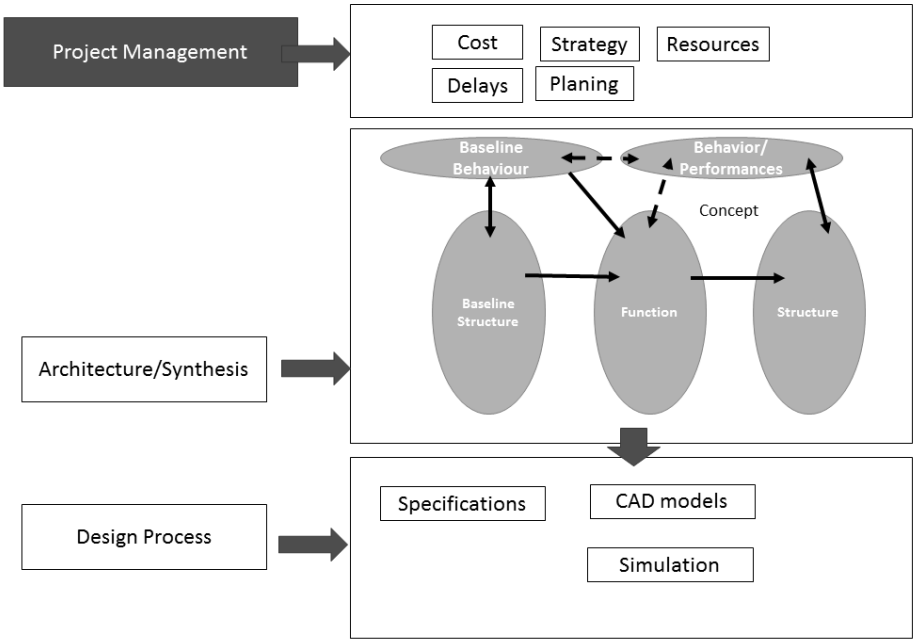
REUSE OF SOLUTION PRINCIPLES



PRODUCT PLATEFORM



FUTURE FLEXIBILITY


Project Management
Cost
Strategy
Resources
Delays
Planing
Baseline Behaviour
Behavior/ Performances
Concept
Baseline Structure
Function
Structure
Architecture/Synthesis
Specifications
CAD models
Simulation
Design Process