Moving Integrated Product Development to Service Clouds in the Global Economy
J. Cha et al. (Eds.)

doi:10.3233/978-1-61499-440-4-564

Set-Based Concurrent Engineering for Early Phases in Platform Development

Christoffer LEVANDOWSKI[a,1], Dag RAUDBERGET[a] and Hans JOHANNESSON[a]
[a] *Chalmers University of Technology*

Abstract. Set-Based Concurrent Engineering (SBCE) is a comprehensive approach to achieve efficiency in product development by providing guidelines to align the development activities. Given a certain range in a design problem, a set of design solutions is defined. Eliminating infeasible regions gradually narrows the set down to a working solution. Similar to platform-based development, SBCE further aims to reuse design knowledge from past development efforts. Although they share the goal of design reuse, there is a fundamental difference between SBCE and platform-based design. The SBCE design process produces one solution while the creation and use of a platform produces a product family. This paper elaborates an approach for platform concept development. It uses set-based concurrent engineering and its principles to develop a platform based on functions and design solutions while preserving the bandwidth. It shows how function-means trees and trade-off curves can represent solution spaces. Further, these spaces are narrowed down to manageable and desirable size to represent a product platform in line with current technology and available manufacturing capabilities. The approach is illustrated with a case from the aerospace industry showing how a manufacturer of parts for a jet engine can develop comprehensive concepts for a platform. The design space is narrowed down using desired bandwidth and compatibility between design solutions. While the approach has proven feasible in the test case, it requires manufacturing and technology development to produce trade-off curves. This implicitly requires technology development and manufacturing development to precede product development. Alternatively, product development, production development and technology development need to be perfectly coordinated in a concurrent manner.

Keywords. *Product platforms, set-based concurrent engineering, concept development, systems engineering*

1. Introduction

The early phases of a development project is characterized by both design freedom and uncertainty. Decisions made early in development are liberated from infringing on investments in expensive manufacturing equipment and frozen designs. This allows for exploring different alternatives at a low cost. However, at this stage, little is known about the design. Set-based concurrent engineering (SBCE) is a comprehensive framework for dealing with multiple alternative solutions throughout all stages of development, until there is only one candidate left. The framework adheres to the Lean Product Development philosophy and contains several basic principles, which has

[1] Corresponding Author.

proven to efficiently provide support for product development in stages such as product planning, concept development, concept screening and detailed design [1].

Yet another approach for efficient development is platform-based design. This approach has received a great deal of attention the last decade as a way to reuse design and manufacturing knowledge. Reuse is a concept which platform-based design and SBCE share [2]. Different things can be included and reused in a platform ranging from reusing assets [3] to the more common reuse of subsystems and interfaces [4]. The intent of that reuse is to form a common structure from which a stream of derivative products can be efficiently developed and produced. This stream of products is commonly referred to as a product family.

SBCE too targets scalability. However, while platform-based design aims to produce a product family, SBCE aims to produce one single product while storing the results from discarded solutions for future use [5].

1.1. Lifecycles of platforms and products

The differences between single product development and platform development manifest themselves in several aspects of the development. Platforms must be prepared with flexibility and maintained for continual use [6]. Developing, using and maintaining a platform can on a high level resemble single product development. Shahin et al. [7] use an established process to develop a platform in much the same way as Ulrich and Eppinger [8] do for single product development, but identify crucial steps where design reuse is facilitated. They mean that reuse can be performed on different levels and that reuse needs to be actively considered. It is however, a platform will need a lifecycle that is separate from the products sprung from it [9]. Pedersen [10] identifies three lifecycle phases of a platform: *preparation*, *execution* and *maintenance*. The platform is both a concept and a design template – thus the design template has to be designed (preparation) and thereafter-derivative products are designed (execution). The template may have to be expanded to align with progressing customer needs (maintenance). The early phases of platform development, i.e. pre-embodiment, fit within the platform preparation stage. The research on detailed processes for such activities is limited. Previous research in the field proposes a general process for preparation which recognizes technology knowledge as an integral part in the preparation process [11]. SBCE provides support for execution [12] and maintenance [13], but covers the preparation only briefly buy providing support for using trade-off curves in platform based-development [14]. Platform concept development and screening is not elaborated further.

The SBCE literature advocates the use of set-based design principles that show some similarity to platform-based designing. Sobek et al. [5] propose a number of principles. These are accounted for in Table 1. Design processes that manage sets of solutions rather than point solutions have shown potential to reduce the probability of design iterations and thus shorten lead times as they increase the chance of finding suitable design solutions [15]. This, as well as the capabilities of narrowing down a set and the common goal of design reuse may prove SBCE to be a good candidate for creating a process or approach for efficient concept development and screening for platform-based development.

Table 1: The principles of SBCE.

Over-all principle	Partial principles
1. Map the design space	• Define feasible regions • Explore trade-offs by designing multiple alternatives • Communicate sets of possibilities
2. Integrate by intersection	• Look for intersections of feasible sets • Impose minimum constraint • Seek conceptual robustness
3. Establish feasibility before commitment	• Narrow sets gradually while increasing detail • Stay within sets once committed • Control by managing uncertainty at process gates

1.2. Bandwidth

A platform is designed to meet a range of customer requirements. This range is created during the concept development and narrowed down to a desirable size in the concept screening. The range may be referred to as bandwidth. Berglund and Claesson [16] introduce bandwidth as systems flexibility, which allows it to be used in a variety of products. Thus, bandwidth may consider the physical and functional properties of a product, such as the range of engines that a car can have, which can be linked to the fulfillment of the range of customer requirements. Consequently, there is a bandwidth of the requirements, and on the design solutions that solves the requirements [17].

In general, there are two approaches to build bandwidth in a platform. First, there are scalable product platforms. Here, the design can be stretched and shrunken to fit specific customer requirements [18]. This is done by manipulating parameters of the design such as the length of a piston or the size of the piston head to achieve different performance.

The other approach is referred to as module-based platforms. A module-based platform consists of a set of interchangeable modules. By changing a module for another, different properties are achieved [19]. For example, changing the lens on a camera may give different focal length without having to change anything on the camera body.

A platform is in most cases either scalable or modular, but never both [20], which may be a result of the prerequisites for execution. The execution of a modular platform assumes that the parameters of each module are fixed, i.e. the geometrical and physical properties of each module are constant [21]. On the other hand, a scalable platform assumes that the architecture is fixed, i.e. no modules will be switched out. Execution through optimization of combined scalable and modular platforms has been tested by Du, et al. [20] using a Stackelberg game approach. However, using a combination of scalable and modular platforms in the preparation phase has not been studied.

1.3. An artifact model for platform-based development

The Configurable Component concept is a platform model that uses both scalable objects and a modular approach. Claesson [22] first introduced the Configurable Component concept as an artifact model to support development of product platforms. The model has been evolved to incorporate manufacturing platforms [23] and to some extent technology platforms [11].

The model represents systems and their sub-systems as objects related to each other. Each system is represented by a Configurable Component (CC), which in turn

can inhabit several design solutions. To fulfill a range of requirements, the configurable component can chose between different design solutions, each of which is scalable. Thus, the model incorporates scalable bandwidth as well as modular bandwidth. The core of the CC is the design rationale. It relates the functional requirements (FRs) to the design solutions (DSs). Each DS may also be affected by a set of constraints (Cs). The enhanced function-means tree as presented Schachinger and Johannesson [24] contains relationships between FR and DS (1:1), and breakdowns of DSs in the next level of FRs (1:n). It also describes the lateral relationships connecting the different branches of the F-M tree. The design rationale is shown in more detail in Figure 1.

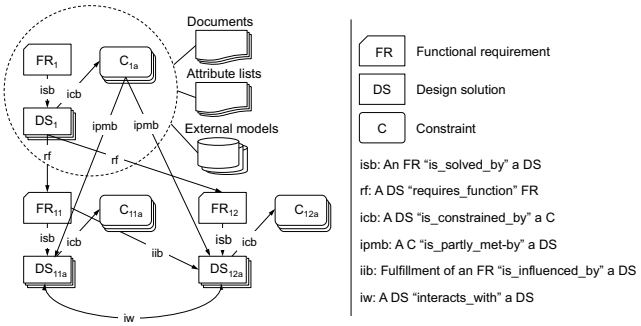

Figure 1: The enhanced function-means tree with the possibility to attach external models and documents describing the Design solutions, functions and constraints

2. Method and Scope

There is a vast literature base on scoping of platforms, and on the platform preparation focused on detailed design, often related to parametric Computer Aided Design (CAD). However, this body of knowledge overlooks the in-between phases of platform modeling pre-embodiment. With proper methods, concept elimination decisions can be based on appropriate information, thus reducing the risk for incorrect decisions.

This paper proposes an approach for modeling platform concepts in early phases and eliminating undesired regions of the design space (section 3). It does so by applying set-based concurrent engineering to create sets of design solutions to cover the bandwidth of each functional requirement. It is illustrated with an example from the aerospace industry (section 4) with information collected over several years' collaboration with the studied company. The results and implications from applying the method are discussed in section 5, and the paper is concluded in section 6.

3. Results - An approach for early phases of platform development

This section describes the proposed approach for early platform preparation. More explicitly, the approach addresses the activities after requirement elicitation and scoping of the platform, and before embodiment through CAD and physics models. Such activities involve functional modeling of multiple design alternatives and elimination of undesired parts of the design space, leading up to a bandwidth for the

platform concept. The proposed approach comprises a bundle of processes, methods and tools.

3.1. The model

The model used to support early platform preparation is based on the CC model. Using F-M-modeling to build up the CC objects from within, the design rationale is modeled in several levels. The levels are defined by the degree of detail, which also determines their proneness to change. Three levels are defined, schematically depicted in Figure 2.

Starting from the top, the majority of functions and sub-functions are located at the *Static and Conceptual levels*. Most of these functions are defining the hierarchy of the system by providing the structure for the underlying functions. The static level consists of the top-level functions that are not going to change in a re-design perspective since they are an integral part of the company's business. An airframe manufacturer rarely initiates the design of next generation products searching for alternative ways of propulsion or producing lift without wings. Therefore, the functions of the upper levels are *static*.

Following the static level, the intermediate functions are realized by a set of alternative design solutions. They effectively drive the platform into different concepts for architectural structures and are therefore called the *conceptual level*. This level can be large and effects all FR and DSs between the Static level and the last level.

The *concrete level* is the last functional description before the allocation of physical components. At this level, the design solutions are more concrete. An object on this level is close to the physical embodiment, and the functions are typically defining features.

The Physical level consists of the components that together build up the system. This level is not part of the functional modeling, except for the links from design solution to physical component.

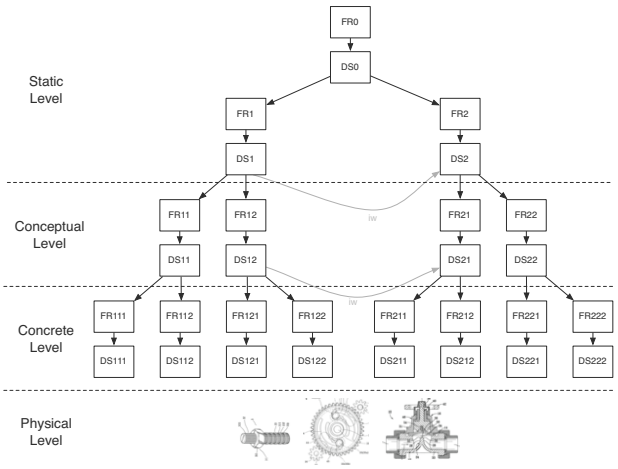

Figure 2: Levels in an F-M tree

3.2. The process

The process of early platform preparation can be supported using the model above. It may be divided into four major activities. The first three, i.e. *generate and structure FRs and DSs, specify solutions* and *narrow down set* are carried out on each level of FRs and DSs. After the F-M tree is ready, the *compilation of bandwidth* starts, applying a bottom up approach.

Initially, a functional architecture is created using function-means modeling. The static level acts as a basis for the creative work. The size of the design effort or change determines at what level to start the modeling. Conceptual development will start with the static level as baseline. Similarly, detailed development will have the conceptual level as baseline.

Each level in the architecture is subject to idea generation on how to solve the function. The ideas are created and collected to cover the entire bandwidth of the functional requirement. For example, an airplane with a functional requirement specified to *generate propulsion* – this is most likely a functional requirement introduced on the conceptual level – may have design solutions such as *jet engine* and *turbo fan engine* to cover the entire desired bandwidth of propulsion power.

Succeeding the functional modeling, the solutions are specified to such a degree that they later can be assessed as feasible or infeasible. Consequentially, this specification job is followed by elimination of undesired solutions from the set. These three steps are iterated on each level of the F-M structure. As the F-M tree is finished, the actual bandwidth is determined by cascading each partial bandwidth upwards to contribute to the total bandwidth of the platform.

3.2.1. Generate and structure solutions

There are several methods for generating solutions. The output from these methods may be a set of possible solutions to a functional requirement. These are organized in relation to what part of the FR-bandwidth they fulfill. At this stage, the accuracy in the determination of each DSs bandwidth will be approximate, due to the lack of detail in the design solutions.

Development rarely starts with a clean sheet, therefore, some branches in the F-M tree may be reused from earlier projects. In case of an expansion of the platform, the majority of the branches will already exist. The relationships between new and old objects will be modeled using the same approach as if there were only new designs, i.e. modeling both vertical and lateral relationships.

3.2.2. Specify solutions

To support accurate elimination of undesired solution, each design solution needs to be specified to such a degree so that they are distinguishable as desired or undesired. In terms of functional modeling, this refers to elaborating on the lateral connections in the model.

On a concept level, specification involves defining the bandwidth using, for example, technology trade-off curves. The lateral relationships may also be specified through trade-off curves describing not only *that* a relationship exists but also *how* the entities affect each other. Specifying design solutions on the detailed level may include parameterized CAD modeling and physical modeling.

3.2.3. Early narrowing of sets

Each FR and DS contributes to the design effort required to mature the platform. The work on each level of FRs and DSs can be extensive, especially if there are many different design solutions per FR, which is why it is important to eliminate undesired designs early, to reduce the workload. The intention of SBCE is to evaluate the architecture of an emergent system before it is elaborated fully. Narrowing the sets is the mechanism to make informed decisions on which alternatives to eliminate and in this phase the goal is to reduce the number of candidate architectures that could fulfill the overall functional requirements.

This approach advocates different ways of narrowing down the set depending on the level of abstraction and the amount of functional coupling between FRs and DSs. The static level does not incorporate any alternatives in design solutions, rendering elimination obsolete.

For the other levels, elimination is possible based on a number of different reasons:

- Elimination based on fulfillment of Bandwidth
- DS elimination based on compatibility with the rest of the platform
- Concept elimination based on compatibility

There are two ways of using the *bandwidth to eliminate* bad or undesired solutions. If there are two interfering requirements, i.e. carry load and stall speed for an airplane, part of the bandwidth of one of the functional requirements may be reduced. This in turn will eliminate several design solutions that previously covered that part of the bandwidth. Which functional requirement to reduce depends on many factors, for example weighting between the requirements etc.

The second case in which the bandwidth may be used to reduce the design set is when design solutions are redundant in their coverage of the bandwidth. Redundant solutions may be eliminated based on how well they relate to other requirements and solutions. This is illustrated in Figure 3.

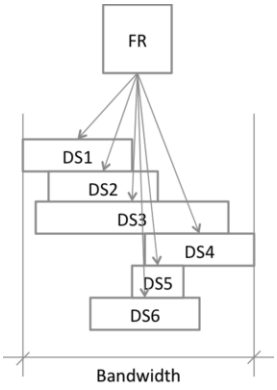

Figure 3: In this picture, there are several redundant DSs. DS2, DS5, and DS6 could be eliminated due o their lack of contribution to the bandwidth (adapted from Wahl and Johannesson [17])

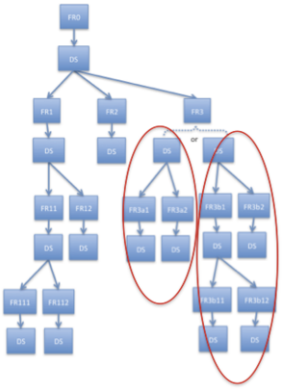

Figure 4: Elimination of branches by eliminating a DS. DSs might not be subject of elimination until they are specified to a certain level.

The initial strategy for elimination is based on *compatibility between Design Solutions*. The solutions are assessed and those found incompatible with the overall system are eliminated [25]. Eliminating a design solution at the Conceptual level effectively prunes a branch of underlying FR and DSs which can be seen in Figure 4.

In theory, all possible variants in the platform are achieved by combining all design solutions on all levels. This creates a huge design space of which great parts will be infeasible. Practically, there are combinations that are not possible, for example using a fly-by-wire steering wheel with a mechanical steering system. Compatibility between solutions can be modeled using the *interacts with* relationship. Thus, these lateral relationships play a major role in defining concepts, and thus eliminating a great number of infeasible designs.

The level on which to start defining concepts is not fixed. Again, Set-based concurrent engineering advocates fact-based decisions. Consequently, the decision on when to start defining concepts (e.g. through morphological matrixes as in [25]) is essentially similar to other elimination decisions. The design needs a level of maturity to irrefutably assure correct elimination. Further elaboration of the F-M tree is needed if unfeasible concepts are not distinguishable, i.e. the knowledge is too low. At the same time, elimination decisions shall cause the design work to progress by reducing the design space early.

Table 2: Elimination on the different levels

Level	Amount of reduction	Method
Static	No reduction	
Conceptual	Reduction of alternate subsystems	Compatibility between DSs
Concrete	Reduction of alternate components	Bandwidth redundancy, bandwidth excessiveness

4. Results - Illustrating Case

To illustrate the approach, an example from the aerospace industry is used. The case company is a supplier for parts of jet engines. Their product offering is broad, covering mechanical design and manufacturing of most of the static parts in a jet engine. Though the company's products can be found in 95% of all civil aircrafts, the volume is still fairly low. Each model is customized for each customer and has a yearly volume of about 400 per model. This makes it impossible to adopt a conventional part-reuse platform strategy. Rather, the reuse can be found on a design solution level. The product studied is a so-called Turbine Exhaust Case (TEC), a static component located at the rear of a jet engine. It has three primary functions that constitute the static level. They are accounted for in Table 3.

Table 3: The static level of the TEC. Here, there are no alternative solutions.

Functional Requirements	Design Solutions
Lead core gas flow	Geometry of inner surfaces
Convey mechanical load between wing and engine	Load bearing structure
Contain disintegrating parts	Circumferential barrier

Figure 5 illustrates a TEC and some of these components in the engine. The customers require customized TEC for their different engine models. Despite the tough individual requirements, the customized variants share some general common design traits driven by the common structure of commercial jet engines: there is an outer ring enclosing the structure, an inner ring connecting to the aft bearing of the engine, and struts connecting the two rings. The conceptual level of the TEC can be found in Table 4.

Table 4: The Conceptual level of the TEC with several DSs for each FR.

Functional Requirements	Possible Design Solutions
Lead core gas flow along outer perimeter	- Geometry of surfaces along outer perimeter - Core gas tube
Lead core gas flow along inner perimeter	- Geometry of surfaces along inner perimeter - Core gas tube
Turn the swirling flow of the Low Pressure Turbine (LPT)	- Geometry of outer surface on vane - Peripheral Bleed-air injection - Spoilers
Convey mechanical loads between TEC and wing interface	- 2- point engine mounting system - 3-point engine mounting system - Line welding
Convey mechanical loads between TEC periphery and engine	- Set of radially extending struts - Set of angular extending struts - Magnetic force conveyer - Asymmetric design direct connection
Connect TEC to outer engine components	- Circumferential flanges

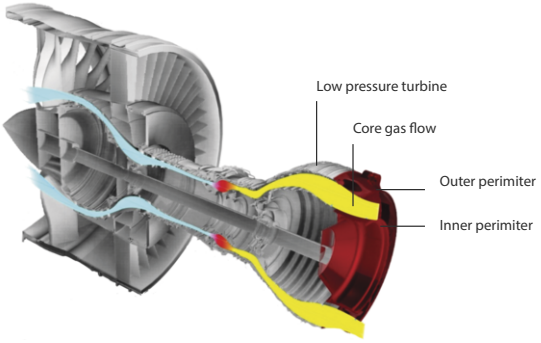

Figure 5: A cut-through of a turbo-fan engine. The TEC is highlighted in red.

Figure 6 illustrates the static and conceptual level that has undergone the first step in the proposed process, i.e. *generate and structure solutions*, resulting in an F-M tree with a multitude of alternative solutions. The next step, *specify solutions*, here means to identify the interactions between different solutions. This is not depicted in Figure 6 due to the obstructing effect in visibility. At this stage, the Combinatory operations reveal 144 different possible concepts, a number that is too large to assess using Computer Aided Engineering (CAE) analysis. In fact, to do so each concept would have to be detailed further. The combinatory explosion would impede the progression of finding a feasible design. Hence, the next step is to eliminate undesired solutions. Given the morphological overview, compatibility of the different architectural options may be identified. Initial strategies for elimination of solutions are described in [25]. Here, a method to evaluate the compatibility between Design Solutions and eliminate those that are not compatible with the overall system is described. In our example, the DSs Magnetic force conveyer and Core gas tube are eliminated due to their incompatibility with all of the other DSs. This action eliminates 117 possible concepts. The pattern of *model and eliminate* is iterated on each consecutive level to produce a manageable design space. Figure 7 describes the breakdown of a DS into a concrete level. On this level, elimination based on the bandwidth may be used. The trade-off curves in Figure 8 describe the bandwidth in the *iw*-relationship between the design solutions in Figure 7.

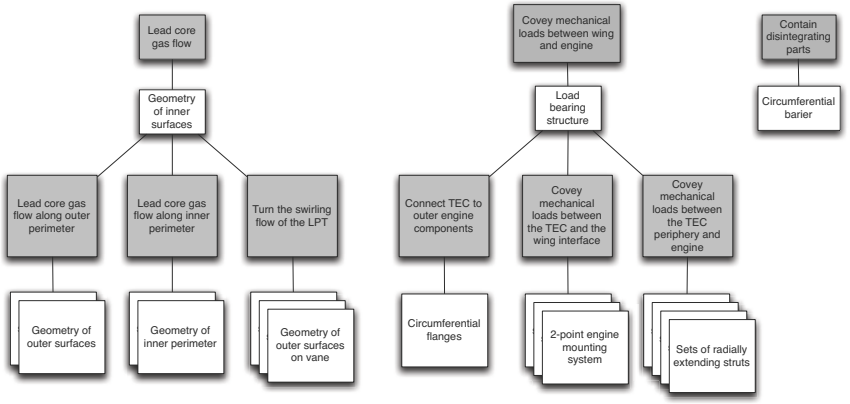

Figure 6: The static and conceptual level of a TEC, adapted from Michaelis, et al. [23]

The TEC typically needs a redirection angle between 0 and 20 degrees. Based on that, profile Z can be eliminated because it is outside the desired bandwidth. Profile Y cannot be eliminated on a redundancy basis, even though profile X performs better in the larger part of the desired bandwidth. Each branch on the conceptual level needs to be developed to a level where the DSs can be mapped to a physical component. As the branches get progressively more detailed, elimination of undesired DSs should get easier. The approach advocates alternating between conceptual elimination and DS elimination using the bandwidth and compatibility with the other parts of the architecture. The resulting set of architectural options constitutes the platform. Each architectural option will still be adaptable on a parametric level, connecting to detailed scalable models such as CAD models of physical components.

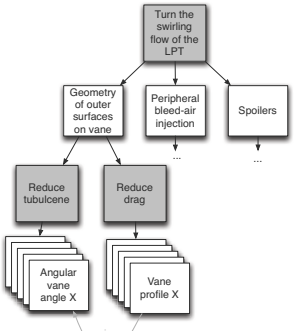

Figure 7: A part of the concrete level of the TEC.

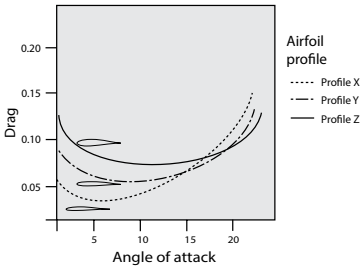

Figure 8: Three different DSs in the concrete level of a TEC. The angle of attack and drag are plotted to define the *iw*-relationship.

5. Discussion

Judging from the literature on SBCE and platform-based design, there has been a void when it comes to detailed processes for efficiently designing platform concepts. The results in this paper aims to fill that void. The presented approach is valid for the case it

illustrates. Though a real development case goes deeper into the architecture, this case serves as an illustrator of the approach suggested. The case was kept to a minimum for pedagogical purposes. Current methods for architectural evaluation in the aerospace industry are based on post embodiment analyses using CAD models and advanced CAE tools. This has proven infeasible for large numbers of architectures due to the massive amounts of data it produces. Adopting simple models, which can reduce the number of possible combinations pre-embodiment helps exploring more alternatives and may ultimately produce a better design.

The approach does not cover the final assessment of bandwidth. Cascading the detailed bandwidth up to assess the upper-level bandwidth is still left for future work. This may help assessing the company offer, further narrowing of undesired solutions not contributing to the customer needs and to identify possible improvement areas. Further, the deliverable to the next phase is in this case a design space of a number of different scalable architectural options. For downstream activities to be able to receive such deliverable and produce a feasible detailed design, ordinary design methods may not be enough [11]. The upstream activities, such as technology development will gain new deliverables, such as trade-off curves. However hard it sounds, SBCE allows for parallel work and close collaboration between departments [1]. In conclusion, an approach like the one suggested in this paper comes with a great deal of change in methods, processes, IT tools and organization.

An essential part in narrowing down a set in SBCE is to gradually narrow down the bandwidth of the requirements [5]. In doing so, the design space that matches the requirements is progressively approaching a single solution. In the case of a platform, it is desired to keep bandwidth and produce the single solutions after configuration. Where the appropriate level of “freezing” the bandwidth is is still subject of future work.

There is software that can accommodate some of the concepts brought up in this paper, for example Inoue [12] and Chin [26]. However, they focus on post-embodiment design. An IT tool for modeling platforms has been developed in connection to this research. This tool can model F-M trees but has yet to be tested for elimination of design solutions.

6. Conclusions

The approach presented here addresses an issue in platform development where little has been explored. There is a vast body of knowledge in platforms and the effects of using them for scalability in production, yet there are few who touches upon how to efficiently and accurately develop select concepts for a platform. This paper contributes with a detailed process, which is illustrated using a case from aerospace industry.

The function-means modeling approach is expanded to fit platform based design, defining three levels of reuse, namely the static level, the concept level, and the concrete level. The static level provides is closely linked to the business goals and does not change between product generations. The conceptual level defines a broad set of concepts aimed to satisfy a range of customer requirements. The concrete level provides detail to the conceptual level, and possesses a bandwidth in both solutions and functional requirements to match the expectations on each concept.

The concept level and concrete level contain a multitude of different design solutions, which are narrowed down throughout the preparation process to a desirable

bandwidth. The elimination of undesired design solutions is done by consulting the bandwidth (redundant solutions and solutions outside of the desired bandwidth are eliminated) and by addressing compatibility between design solutions. The elimination of undesired solutions help in design space manageable in further design efforts.

The final platform concept will act as input for detailed design using CAD models and Finite Element Analysis. The final bandwidth of the platform relates to the bandwidth of each design solution. How to cascade the bandwidth upwards to assess the fulfillment of business goals is still a subject of future work.

References

[1] Liker, J.K., Sobek, D.K., Ward, A.C. and Cristiano, J.J. Involving suppliers in product development in the United States and Japan: evidence for set-based concurrent engineering. *Engineering Management, IEEE Transactions on,* 1996, 43(2), 165-178.

[2] Levandowski, C., Michaelis, M.T. and Johannesson, H. Set-Based Development Using an Integrated Product and Manufacturing System Platform. *Concurrent Engineering - Research and Applications,* 2014, (accepted)(

[3] Robertson, D. and Ulrich, K. *Planning for Product Platforms*, 1998 Vol.39).

[4] Meyer, M.H. and Lehnerd, A.P. *The Power of Product Platforms: Building Value and Cost Leadership*, 1997 (Free Press, New York, New York, USA).

[5] Sobek, D.K., Ward, A.C. and Liker, J.K. Toyota's principles of set-based concurrent engineering. *Sloan Manage Rev,* 1999, 40(2), 67-83.

[6] Pedersen, R.: Product Platform Modelling - Contributions to the discipline of visual product platform modelling. Department of Management Engineering. Technical University of Demark, Lyngby, Denmark (2009)

[7] Shahin, T.M.M., Andrews, P.T.J. and Sivaloganathan, S. A Design Reuse System. *Proceedings of the Institution of Mechanical Engineers Part B - Journal of Engineering Manufacture,* 1999, 213(6), 621-627.

[8] Ulrich, K.T. and Eppinger, S.D. *Product Design and Development*, 2012 (McGraw-Hill/Irwin, New York).

[9] Wortmann, H. and Alblas, A. Product platform life cycles: a multiple case study. *International Journal of Technology Management,* 2009, 48(2), 188-201.

[10] Pedersen, R.: Product Platform Modelling - Contributions to the discipline of visual product platform modelling. DTU Management, Vol. Ph.D. Thesis. Technical University of Denmark, Lyngby, Denmark (2009)

[11] Levandowski, C.E., Corin-Stig, D., Bergsjö, D., Forslund, A., Högman, U., Söderberg, R. and Johannesson, H. An integrated approach to technology platform and product platform development. *Concurrent Engineering: Research and Applications,* 2013, 21(1), 65-83.

[12] Inoue, M., Nahm, Y.E., Okawa, S. and Ishikawa, H. Design Support System by Combination of 3D-CAD and CAE with Preference Set-based Design Method. *The Journal of Concurrent Engineering: Research and Applications,* 2010, 18(1), 41-53.

[13] Michaelis, M.T., Levandowski, C. and Johannesson, H.: Set-Based Concurrent Engineering for Preserving Design Bandwidth in Product and Manufacturing System Platforms. Proceedings of ASME IMECE, San Diego, California, USA (2013) Paper No. 63624

[14] Levandowski, C., Forslund, A., Söderberg, R. and Johannesson, H.: Using PLM and Trade-Off Curves to Support Set-Based Convergence of Product Platforms. 19th International Conference on Engineering Design – ICED 2013, Seoul, Korea (2013)

[15] Shahan, D. and Seepersad, C.C. Implications of Alternative Multilevel Design Methods for Design Process Management. *Concurrent Engineering,* 2010, 18(1), 5-18.

[16] Berglund, F. and Claesson, A.: Utilising the Concept of a Design's Bandwidth to Achieve Product Platform Effectiveness. Proceedings of ICED 2005, Melbourne, Australia (2005) Paper No. 377.346

[17] Wahl, A. and Johannesson, H.: Managing Design Change in Configurable Component Based Platforms. Proceedings of the 8th Biannual Conference NordDesign 2010, Vol. Vol. 2, Gothenburg, Sweden (2010) 413-423

[18] Simpson, T.W. Product platform design and customization: Status and promise. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing,* 2004, 18(1), 3-20.

[19] Gonzalez-Zugasti, J.P. and Otto, K.N.: Modular platform-based product family design. Proceedings of ASME DETC. Citeseer, Baltimore, Maryland, USA (2000) Paper No. 14238
[20] Du, G., Jiao, R.J. and Chen, M. Joint optimization of product family configuration and scaling design by Stackelberg game. *Eur J Oper Res,* 2014, 232(2), 330-341.
[21] Fujita, K., Sakaguchi, H. and Akagi, S. Product variety deployment and its optimization under modular architecture and module commonalization, in *Proceedings of Proceedings of the 1999 ASME design engineering technical conferences,* 1999, pp. 12-15
[22] Claesson, A.: A Configurable Component Framework Supporting Platform-Based Product Development. Department of Product and Production Development, Vol. Doctor of Engineering. Chalmers University of Technology, Gothenburg, Sweden (2006)
[23] Michaelis, M.T., Johannesson, H. and ElMaraghy, H.A. Function and Process Modeling for the Co-Development of Products and Manufacturing Systems. *Journal of Manufacturing Systems,* 2014, in print),
[24] Schachinger, P. and Johannesson, H. Computer Modelling of Design Specifications. *Journal of Engineering Design,* 2000, 11(4), 333-353.
[25] Raudberget, D.: Enabling Set-Based Concurrent Engineering in Traditional Product Development. Proceedings of ICED 2011, Copenhagen, Denmark (2011) Paper No. 68.61-145
[26] Chin, K.-S., Chan, A. and Yang, J.-B. Development of a fuzzy FMEA based product design system. *Int J Adv Manuf Technol,* 2008, 36(7-8), 633-649.