Honda Research Institute EU
Honda Research Institute Europe GmbH
https://www.honda-ri.de/

Addressing Complex Engineering Systems: The Value Of Evolvability

Stefan Menzel, Timo Friedrich

2019

Preprint:

This is an accepted article published in IEEE Symposium Series on Computational Intelligence. The final authenticated version is available online at: https://doi.org/10.1109/SSCI44817.2019.9002695 Copyright 2019 IEEE

Addressing Complex Engineering Systems: The Value Of Evolvability

Stefan Menzel, Timo Friedrich
Honda Research Institute Europe
Offenbach/Main, Germany
{stefan.menzel, timo.friedrich}@honda-ri.de

The article has been accepted for publication in the IEEE Symposium Series on Computational Intelligence.

Copyright notice:

©2019 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Addressing Complex Engineering Systems: The Value Of Evolvability

Stefan Menzel, Timo Friedrich
Honda Research Institute Europe
Offenbach/Main, Germany
{stefan.menzel, timo.friedrich}@honda-ri.de

***Abstract*—While engineering systems in the automotive development process not only become increasingly complicated, the necessary consideration of uncertainties shifts the focus furthermore towards system complexity. For an efficient decision-making flow, i.e. making the right decision at the right point in time, it is important to consider simultaneously both engineering paradigms – on the one hand to accurately and computationally efficient solve the complicated part and, on the other hand, to smartly handle the complex part of a system – as well as their interactions. Smart handling in the sense of allowing system openness, discovery and system preparation for solution portfolios is realizable by defining prerequisites and qualities of a system. Evolvability as one of these qualities is a major actor in complex systems engineering since it strengthens a system's state of readiness, the adaptation capability of a system towards structural changes over time.**

In the present paper, paradigms and approaches for complicated and complex systems engineering are reviewed and discussed from a practitioner's point of view. The concept of evolvability and its practical relevance are highlighted by research examples on the evolvability of deformation methods as representations in evolutionary design optimization as well as on platform systems with respect to evolvable prototyping.

Keywords—systems engineering, complex systems, evolvability, dynamic environments, adaptation potential

I. INTRODUCTION

Tackling complex engineering systems systematically in the right way, i.e. learning how to efficiently handle and steer a system's development capability rather than only finding one optimal system solution for a pool of constraints, requirements and functional specifications, is of major importance in current product development with even more relevance in future applications. Dealing with changes in the environment or in the system is not only a singular moment of occurrence, but these changes have to be expected, welcomed, actively integrated and maybe even stimulated. In the idea of systems engineering, this step marks the transition from systems engineering of complicated systems to systems engineering of complex systems. Following the consensus of the International Council on Systems Engineering (INCOSE) fellows, systems engineering "integrates all the disciplines and specialty groups into a team effort forming a structured development process that proceeds from concept to production to operation. Systems engineering considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs." [1]. A system is referred to as a combination of different components that in the sum produce results superior to the results of each single component.

In the spirit of complex systems engineering, "'becoming' is 'being'" [2]. The approach needs to take place in an open-end discovery process which allows a system to comprise a variety of potential solution options – rather than one optimal final solution – which undergo a constant structural adaptation to strive for further novel, sometimes surprising solution options. Optimization under uncertainty, system robustness and flexibility are important key qualities needed in the complex systems engineering approach. From this perspective, the complex engineering approach is rather not based on classical engineering techniques for solving a pre-specified task, but it is more a process to generate an environment and to design working conditions where certain system qualities, such as self-organization and structure adaptations are stimulated. The process can be well summarized under the phrase of "learning to dance" [3] with a complex system that is substantially different from purely solving it.

Motivated from biology and evolution processes, evolvability is one of such system key qualities. In a general sense, evolvability determines the level of a system to offer solution portfolios for environment changes to minimize the chances of dead-ends, and the capability to adapt the system structure. We also argue that the notion of time is a highly important guiding principle, especially with respect to decision-making. Assuming a given final product delivery date fixed by the sales team, the optimal decision-making process for optimal product performance considers to keep a system as flexible (evolvable) as long as possible while having in mind the time which is required to converge the current flexible system state to final optimal solutions, see Fig. 1. Of course, for real world engineering applications the process is even more complicated since final dates typically shift during the development process resulting in uncertainties towards timing.

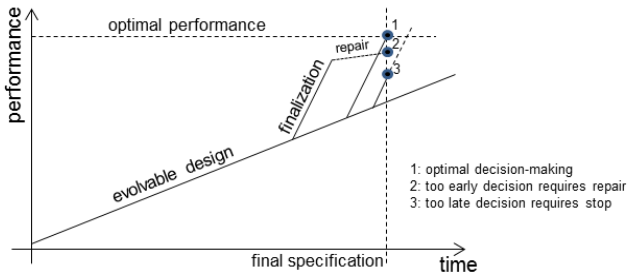

Fig. 1. Simplified engineering decsision-making.

On a very simplified level, one may assume that development teams strive for a product solution required at a specified deadline, which may be the final deadline but also intermediate deadlines between single teams. These teams typically try to keep as many options as possible to react to potential changes, which may be expected by product specification updates during development time. It can be envisioned that in the ideal case a team would try to keep the system as open as possible as long as possible knowing the optimal point in time when to switch to converging the flexible system to the solution(s) with the best performance. In case the point in time would be chosen too early the system would not reach the best performance but teams would use the remaining time to repair and improve the product as much as possible. Or, if the point in time was chosen too late, the system cannot fully converge. In both scenarios, the idealized best performance can usually not be achieved.

In the light of the guiding principles outlined above, a classification of systems engineering for complex and complicated problems by type and not by degree, as proposed in [3], provides the chance for coexisting engineering paradigms and approaches. On the one hand complex systems engineering targeting systems with maximum evolvability and on the other hand classical systems engineering for complicated problems targeting fully converged and customized products. Classical systems engineering is considered here as a process to produce efficient and reliable products that meet constraints, standards and performance requirements in pre-specified situations.

As also discussed later, evolvability can be identified and utilized on different levels in the digital development process. Taking evolutionary optimization as example, researchers typically target to design an optimization method as general and flexible as possible for a variety of problem classes, e.g. by designing a pool of mathematical mutation or selection operators. On another level, these operators come along with parameters that automatically adapt during a single optimization run to allow for optimal exploration and exploitation behavior, as e.g. the smart covariance matrix adaptation in CMA-ES [4].

In the present paper, our findings for the utilization of evolvability in the digital development process are summarized along two exemplary applications. The first application focuses on the design of efficient geometric representations for 3D shape optimizations. Assuming the designer/engineer decided on using shape morphing methods, which is a very common 3D representation within engineering optimization, one may ask from a general perspective which of the various shape morphing techniques (some of them explained later) would allow for the highest degree of evolvability. By comparing different high-level characteristics such as adaptability, robustness, precision and efficiency insights are provided on favorable methods. In addition, after deciding on which variant of shape morphing is utilized in the design optimization, it is important to research how to build the most efficiently set-up, which typically means how to choose the optimal number of parameters and how to distribute the object handles on the geometry. The second part of section III provides insights on evolvability-inspired criteria for calculating a Pareto front of potential parameter distributions from which the user can select the optimal set-up. In a second application, a software platform is introduced which is utilized to optimize the platform of an engineering benchmark. Platforms are important concepts in engineering since they should provide a very flexible base system (which should be highly evolvable) allowing to rapidly optimize for specific product implementations along the chosen product line. The optimization demonstrates that efficient platforms allow for the fast adaptation of a new product if the new product specifications are similar to the existing platform.

In summary, this paper reports on our findings for practical ways of preparing systems to find optimal solutions in dynamic engineering environments. In Section II, more insights on complex systems and the differences to complicated systems in an engineering context are given. Section III highlights that high-quality representations increase evolvability in a general sense and presents a method for an automatic set-up of shape morphing methods for evolutionary design optimization. An application from the domain of product platform family optimization is provided in more details in Section IV. Section V concludes the paper.

II. Complex Systems Engineering

A. *Definition and Properties*

In this section, ideas from literature on definitions for complexity with respect to complex systems/problems, especially complex engineering systems, are given. The terms complexity and complex systems have manifold notions depending on the field of research and application under consideration. Additionally, often a misunderstanding occurs by mixing (very) complicated systems with complex systems [2], [3]. A very basic definition for complex systems is given in literature “in terms of a number of elements of having diverse, variable behaviors that interact to produce the behavior of the whole” [5]. To add a missing piece, in this context authors commonly agree on the attribute of complex systems to be a result of networks of multiple interacting systems which have to be considered as a whole, and comprise the characteristic of self-organization [2], [3]. Besides more general issues as uncertainty, robustness and flexibility [6], system properties as non-linearity, chaotic behavior, many feedback loops, scale-free behavior and network effects are stated [5]. Furthermore, especially emergence resulting from the self-organization process is stressed as key property of complex systems. Complex systems are changing and adapting over time due to the changing environment, and importantly, the direction of a present system change directly influences the future direction of change in the environment.

In [3], a difference is outlined between complicated and complex systems for social systems which basically can also be transferred to engineering systems. Complicated problems can be partitioned into subsystems which can be tackled individually, deliver proportionate output and result in reproducible solutions. In contrast, complex systems/problems require a holistic approach, i.e., interactions of subsystems need to be carefully taken into account, the problem cannot be solved once and forever, and interventions over time have to be considered. The author in [3] argues that this difference is a

difference in type and not only degree, which as a consequence allows a system to be both, complicated and complex. Complicated systems can be understood through their structural decomposition and –theoretically– fully modeled. Complex systems on the contrary are understood through functional analysis. Modelling complex systems is typically incomplete and diverges over time. The main reason is that complex systems have the capacity to change over time. If in this change process some positive value is assignable to the consequences it may also be interpreted as creativity [3].

Systems engineering of complicated systems, e.g. products such as microprocessors, takes a reasonably portion in the daily business of product development where accurate models are built to evaluate the performance of a (part of the) product for a pre-specified scenario. Especially the rise in the capabilities to model subsystems in more detail and to execute more accurate simulations in faster computational time allow the broadening of the system perspective. Hence, a larger degree of the design space is explorable which therefore allows assessing more knowledge and information of the problem. This brings up the idea of focusing on the holistic system, i.e. with higher focus on the interactions and interrelations of subsystems, which requires different and more advanced engineering approaches to come up with good solutions. Qualities such as system adaptation potential that needs to be handled on a representation level as well as on an optimization algorithm level and the capability of self-organization are important ingredients for a complex systems engineering approach.

Following this idea, it seems natural to consider principles from biology in the search of approaches to deal with and to engineer complex systems. Utilizing evolutionary processes for introducing system variations and parameter selection allows *situated design* in the context of the actual task. The structure of a complex system does not reflect static optimality but an evolving fitness, which is constrained by a dynamic, co-evolving space of possibilities [2]. A guiding principle for the evolution success is *evolvability*. Evolvability is an evolved quality and is specified as "the ability of the configuration space (in this case, the space of genotypes and phenotypes) to produce an endless supply of viable configurations with remarkably few obvious dead-ends" [2]. This definition contains many similarities to a definition where "evolvability is considered in the sense of the capacity of a system to produce favorable phenotypic variations of a design within a moderate number of generations while avoiding non-feasible mutations" [7] which has been used to optimize the structure of representations. Complex systems are thought to "share structural properties that affect how easily they can be modified to change what they do" [8]. Along this line, following the spirit of evolvability two further overarching design principle for complex large-scale engineering systems, *sustainability* [9] and *robustness* – especially robustness over time [10] – can be finally touched. In the next section, ideas for a complex systems engineering paradigm in contrast to a classical approach are summarized.

B. Paradigm for Complex Systems Engineering

The classical engineering process, a process for solving complicated problems, usually starts with a functional specification of the tasks that a system is expected to perform. Constraints and tolerances are given to detail the conditions of the operating environment. Based on the specifications, the design process is carried out and the system components are sequentially or simultaneously created. The testing and validation is afterwards done by performing system tests for a given set of specified conditions which mimic the real world environment. After finalization, the product layout is handed over to manufacturing for mass productions and distribution.

The classical engineering process targets the search for a single solution, which provides a unique concept for a pre-specified problem. A problem is stripped down in a top-down manner to system parts and interactions, which are well understood and accurately modeled during development. The engineers and designers look for well-behaved systems, which comprise qualities such as predictability, stability, reliability, transparency, controllability, precision and optimality [2].

However, in the defense of classical engineering it can certainly be assumed that based on their expertise and experience engineers and designers usually try to generalize their developed methods and solutions to cope with uncertainties emerging in the development process, which is probably intrinsic to human nature. A guiding constraint is nevertheless given by the maximum time budget for solving the problem, which, of course, sets the major focus point to guarantee the certification of a product functionality in time by meeting industry standards, norms and regulations provided e.g. by DIN, TÜV, NHTSA, IIHS, or EURO NCAP for automotive applications.

In contrast to the classical engineering process, complex systems engineering shifts the focus point to develop systems comprising the capability to adapt, evolve, innovate or even grow after release. The boundaries of a closed environment are crossed and complex systems are expected to function in open environments with unforeseeable contingencies [2].

Following the importance of adaptation potential and self-organization for complex systems, desired key concepts in a complex systems engineering paradigm are based on the ability that local actions result in globally advantageous consequences. Systems are enabled to reconfigure themselves to handle novel situations with incomplete information, they expect the unexpected. An important aspect in this paradigm idea is modularization, which is the availability and occurrence of modularity resulting in a diversity of module and/or process combinations with identical consequences [2], [11]. This links to the idea of neutral networks in biology and adds to redundancy and degeneracy on all levels [12]. Existing modules and processes are reused with minor modifications, so that based on the variety within the system multiple approaches can be tried out and progressive improvement can be achieved. In biology, this concept is called exaptation, i.e. to explore for new designs and functionality by recombination of genes and modules, and exploit or fine-tune a system based on existing genes [5]. The combinatorial explosion may be seen as a chance providing a higher probability that a system has solutions already available for suddenly occurring changes. In the ideal case, a robust structure evolves which prevents

system-wide fatal breakdowns, providing multiple interaction variants of systems elementary components [2].

A very good summarizing phrase for the fundamental issue of designing complex systems is given by "the need for solution-rich configuration spaces" [2]. Consequently, the challenges for complex system engineering are different to the ones in the classical engineering approach. On an abstract level, the idea to obtain specific global functionality from local processes may be achieved by the coordination of interactions, the implementation of analogies from complex biologic systems as well as selective plasticity, a combination of learning and selection mechanisms [13]. Finally – and maybe most challenging –, the development of performance metrics to evaluate the quality of complex systems needs to be addressed [2]. A novelty-driven approach has been proposed in [14].

In the next sections, we provide an overview on current research for embedding evolvability in practical engineering systems as preparation mechanism in dynamic environments.

III. Evolvability of Representations

Geometric representations play a major role in the vehicle development and optimization process. Based on the chosen representation the geometric parameters are defined which tremendously influence the structure of the search space and efficiency of the search. The quality of representations in general is characterized by completeness, i.e. all feasible solutions should be reachable, regularity, i.e. no solutions should be favored by the representation, and strong causality, i.e. the relations of induced variations on parameters (genotype) and resulting variations of the shape (phenotype) should be correlated.

With respect to evolvability, representations are discussed from two perspectives. On the one hand, a representation is evaluated from a general perspective on its level of capability and efficiency to realize changes as well as to be changed, i.e., here, different *types* of representations are compared against each other. On the other hand, a representation is evaluated on the most promising set-up to allow design changes in uncertain environments during the development or optimization process, i.e., here, different *set-ups* within one single representation type are compared against each other. Next, both perspectives are explained on deformation methods for simulation-based evolutionary design optimization in more detail.

A. Increased Boundaries: Maximization of Search Space

In the automotive development, the modelling and simulation of physical processes to evaluate and compare different design options is common practice. For the analysis of internal fluid dynamics, e.g. combustion or catalyst efficiency, or external fluid dynamics, e.g. drag efficiency, as well as structural efficiency, e.g. component stiffness, 2D/3D simulation software tools are usually applied which solve the computational fluid dynamics (CFD) and finite element (FE) differential equations. These simulation tools rely on a high-quality discretization of the design surface, which is often given in CAD format, e.g. NURBS surfaces, and the simulation space, e.g. the space around a car or inside a component. The discretization process results in different

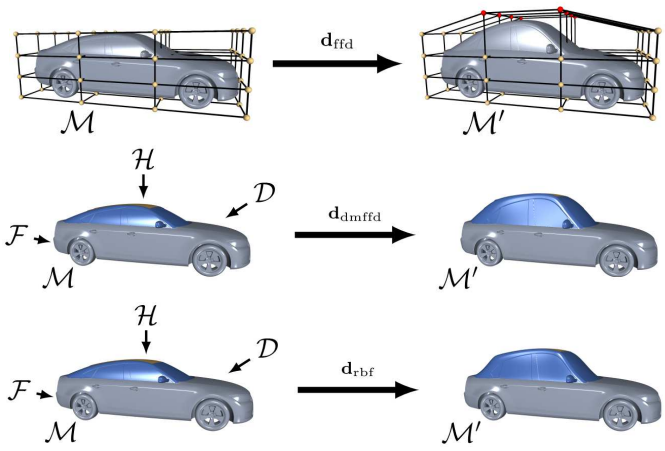

Fig. 2. FFD (top), DMFFD (middle) and RBF based deformation (bottom) of the DrivAer [24] model, taken from [15]. H denotes a handle region (gold), D denotes a deformable region (blue), F denotes a fixed region (grey) and M, M´ denote the initial and deformed model.

object representations such as triangular or quadrilateral surface meshes or volumetric meshes, i.e. point clouds, which possess problem dependent mesh topologies. During the design optimization step, the object geometry needs to be modified for a large number of parameter suggestions and each design variation requires an update of the numerical grid.

Space deformation methods [15], [16] comprise high potential for these kinds of simulation-based optimizations due to their capability to represent changes on a baseline design rather than the design directly and to deform the embedding space around or within an object (Fig. 2). State-of-the-art techniques, such as free-form deformation (FFD), direct manipulation of free-form deformation (DMFFD) and radial basis function based deformation (RBF) comprise the advantage to modify design surface and numerical grid simultaneously and avoid a costly, sometimes manual, remeshing process. These methods allow a fully automated evolutionary design optimization workflow and are very robust against geometry defects such as badly shaped triangles, non-manifold mesh cells or disconnected cells. The techniques require a baseline geometry and numerical grid and differ in the function *d*, which blends the point coordinates of the original geometry to the coordinates of the updated geometry. From the user interface point of view, a difference also exists in the process of design interaction. While FFD deformations [17], [18] are realized indirectly via control point modifications, DMFFD [19] and RBF [20]-[23] provide object handles, which can be directly defined on the design surface. In the following, more details are given for the different deformation methods.

FFD is realized by constructing a lattice of control points around a given object and deforming it using trivariate tensor-product Bezier or B-spline functions. For targeting local support and computational efficiency typically cubic B-splines are utilized in practice. At first, the coordinates of the surface and/or volume mesh are calculated in spline parameter space. In a second step, positions of control points are manipulated, e.g. by an evolutionary optimizer, and the mesh coordinates are updated using the modified control point positions.

DMFFD also relies on a control point lattice but in contrast to FFD, deformations are realized by modifying multiple object handles, which positions are specified by the user directly on the design surface mesh. The deformation is computed in a two-step process. At first, the control point lattice is calculated to reflect the movement of the object handles with minimum error, e.g. by using a computationally efficient least squares approach such as singular value decomposition (SVD). In a second step, the modified surface and/or volume mesh is calculated using the updated control volume. It has been exemplarily shown that DMFFD offers advantages over FFD in an aerodynamic design optimization scenario because of the more efficient search space organization. Furthermore, DMFFD is more sensitive to violations of constraints [25].

RBF based space deformations improve upon FFD and DMFFD in two significant aspects: First, the adaptation to introduce additional degrees of freedom is highly flexible due to its kernel-based nature without the need to maintain a complicated control lattice structure. Second, by choosing an adequate basis function the RBF deformation can directly include the minimization of a physically inspired energy (fairness energy) and realize physically plausible and smooth deformations. Choosing a triharmonic function $\phi(r) = r^3$ penalizes third-order derivatives and is globally C^2 smooth [26]. In contrast to FFD and DMFFD, the fixed and deformable design regions are specified by the user flexibly on the object. Design variations are realized by modifying user chosen handle regions by, first, solving a linear system of equations to calculate the coefficients of the RBF deformation maintaining exactly the constraints. Second, the object is updated by evaluating the RBF deformation for each vertex.

In [15], space deformation methods are compared according to performance, robustness, quality, adaptivity and precision. We assume that maximizing these qualities improves the theoretical distance to the search space boundary by minimizing the probability of generating invalid shapes in an evolutionary design optimization, thus increasing *evolvability* as defined above. For performance, i.e. computational time and robustness, FFD and RBF are on equal level, DMFFD lacks in both criteria because of the computational effort to solve the SVD and potential numerical instabilities of the SVD eventually resulting in huge distortions of the control volume. Differences between both, FFD methods and the RBF based method become more obvious when comparing deformation quality, adaptivity and precision. Deformation quality is evaluated by following the principle of simplest shape, i.e. the deformation function should be smooth, fair and physically plausible.

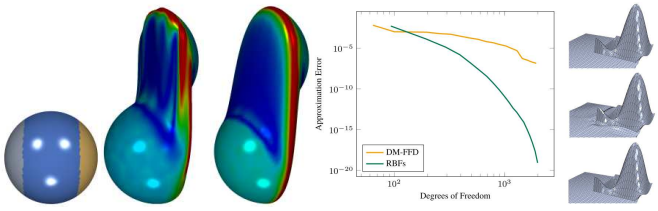

Fig. 3. Left: Mesh smoothness based on curvature. Right: Adaptive refinement benchmark results, taken from [15].

For a given sphere deformation the mesh smoothness based on curvature has been computed and is visualized in Fig. 3. Next to the sphere, the DMFFD (729 control points) and RBF deformation (792 kernels) are depicted. High curvature is indicated by red colors. The lattice based methods suffer from aliasing artifacts whereas the smoothness of RBF profits from the built-in minimization of physically inspired (fairness) energies.

Adaptivity has been compared on a benchmark function [15], which comprises global and local features. For increasing degrees of freedom of a base deformation by inserting control points (DMFFD) or function kernels (RBF) the approximation error is calculated. The superior performance of the RBF deformation, i.e. lower approximation error with increasing number of points/kernels, is depicted in Fig. 3, right.

In the context of simulation based design optimization, the quality of the CFD/FE numerical grid strongly influences the quality and convergence of the computational performance evaluations. For comparison, the roof area of the DrivAer model [24] has been deformed and re-meshed using the OpenFOAM [27] snappyHexMesh tool. OpenFOAM is a CFD solver used for aerodynamic simulation in industrial design optimization. The results assessed by the OpenFOAM checkMesh tool show the superior quality of RBF deformations while the lattice based deformation methods produce numerical cells of weak quality (Table I and Fig. 4).

A final assessment is given in Table II. RBF deformation provides the best results and is strongly recommended for simulation-based optimization. Arguing from an *evolvability* perspective, RBF deformations comprise the highest potential to successfully generate valid designs within the mutation step

TABLE I. RESULTS REPORTED BY OPENFOAM'S CHECKMESH. SUCCESSFUL TESTS ARE INDICATED BY A CHECK MARK, WARNINGS BY !, AND ERRORS BY X. NUMBERS ARE GIVEN FOR THE WORST QUALITY ELEMENT IN THE MESH, TAKEN FROM [15]

	Aspect Ratio	Cell Orthogonality	Face Skewness	Face Pyramids
Original	6.9 ✓	64.7 ✓	3.4 ✓	✓
RBF	6.6 ✓	68.6 ✓	3.7 ✓	✓
DM-FFD-10	7.0 ✓	71.3 !	3.6 ✓	✓
DM-FFD-15	7.0 ✓	70.7 !	3.4 ✓	✓
DM-FFD-25	2.5e+195 ✗	179.7 ✗	1031.8 ✗	✗

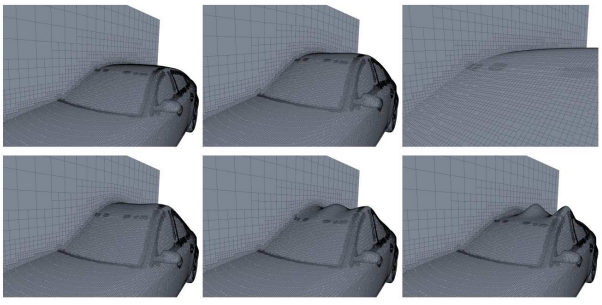

Fig. 4. Cut-view and car surface patch of the resulting volume mesh after deformation. Original (top left), RBF (top center), zoom to the surface (top right), DM-FFD (bottom row). In case of DM-FFD, the results for control grid resolutions 10^3, 15^3, and 25^3 are shown, taken from [15].

TABLE II. SUMMARY. FOR EACH BENCHMARK TEST AND DEFORMATION METHOD, NEGATIVE (–), NEUTRAL (O), OR POSITIVE (+) ASSESSMENT IS PROVIDED, TAKEN FROM [15].

	Performance	Robustness	Quality	Adaptivity	Precision
FFD	o	+	o	–	–
DM-FFD	–	o	o	–	o
RBF	o	+	+	+	+

of an evolutionary design optimization and react to dynamic changes because they provide the highest deformation quality and precision while providing a high capacity to be adapted if environment changes require more local or spatially different deformations. FFD is recommended if fast and simple conceptual design exploration should be robustly achieved.

B. *Automated Construction of Deformation Set-ups for Evolutionary Design Optimization based on Evolvability*

In the previous section, we qualitatively compared different implementations of shape deformation techniques from a general perspective. We argued that a higher level of computational performance, robustness, deformation quality, adaptivity and precision increase the evolvability of a representation in general because more valid design proposals are reliably generated. As a consequence, it is now possible to choose one of the above evaluated deformation methods and study the effects of different set-ups, i.e. the number and distribution of control points (CPs) for FFD or kernels (KEs) for RBF, when utilized to initialize an evolutionary design optimization for a specific problem with uncertain solution space. Variations in the set-ups have substantial impact on the size and structure of the search space. Generally, it can be assumed that more equally distributed CPs/KEs are favorable for exploring the search space because dense distributions increase the risk of numerical instabilities and, consequently, invalid surface meshes. In contrary, adapting CPs/KEs locally in regions where the highest fitness gain is expected results in more dense distributions, which favor exploitation since less optimization steps, i.e. fewer CPs/KEs movements, are expected to find the (local) optimum. From a practitioner's point of view, quantifying exploration and exploitation for a representation would offer the valuable chance to computationally optimize the set-up of deformations, thus releasing the user from a manual and often very tedious set-up process.

In [28] and [7], the effects of different initial FFD control point distributions in evolutionary design optimization have been studied showing that considering evolvability in the set-up improves convergence time and solution quality. As a side effect, the advantage of the CMA-ES as example for an adaptive evolution strategy was also observed repairing weaker CP set-ups during the course of optimization by adjusting mutation step size and search direction. Recently, we addressed the idea of utilizing evolvability as a concept for setting up deformation methods for evolutionary design optimization in dynamic environments [29]. For this, linear deformation representations like FFD, DM-FFD and RBF are typically written in the following form:

$$M' = M + Up \tag{1}$$

M is the initial model mesh, M' is the deformed model mesh, p contains the modifications of CPs/KEs and U is the deformation matrix. Based on these matrices three qualities, namely variability, improvement potential and regularity [30] have been mathematically defined. Conceptually, variability V and improvement potential P can be interpreted as exploration and exploitation capability while regularity R refers to convergence speed. Variability V provides a measure on how balanced CPs/KEs are distributed among the surface mesh by:

$$V = (X_d - h_{X,C})/X_d,\ V \in [0,1] \tag{2}$$

$$h_{X,C} = sup\ min\ ||x\text{-}c||,\ x \in X,\ c \in C \tag{3}$$

where X are all vertices of the surface mesh, C are all RBF kernels and X_d is the distance when placing only one kernel on the mesh with the farthest distance to all other kernels. V equal to 1 would imply a very balanced kernel distribution ideal for exploration, which would be e.g. achieved by Lloyd sampling.

Improvement potential on the contrary considers gradient information g, which estimates future variations, and includes uncertainties. These gradients may result from data analytics of historic data, prediction models or designer's educated guesses and allow for rapid exploitation of the search space. According to [29], P is calculated using the identity matrix I by:

$$P(U) = 1 - ||(I - UU^{+})g||^2,\ P \in [0,1] \tag{4}$$

Based on V and P it is possible to calculate multiple deformation set-ups, which are a trade-off between exploration and exploitation. As example, target shape matching scenarios as replacement for costly aerodynamic simulations have been used where a plane (Fig. 5, left) is fitted to a target plane with a step and a bump (Fig. 5, right). As a side note, 3D face matching has also been studied in [29], [30] as extended scenarios but because of limited space it is not discussed here. However, the results on face matching have similar quality.

To get an overview on the solutions 51 single-objective optimization runs using CMA-ES have been carried out. The parameters are the coordinates of the RBF kernels and the fitness value is computed by a weighted sum of P and V. The results are depicted in Fig. 6 in a multi-objective plot. CMA-ES has been used because of faster convergence compared to a full multi-objective optimization using e.g. NSGA-II [30].

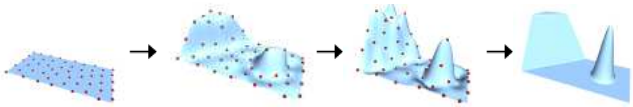

Fig. 5. RBF based target shape matching, adapted from [29].

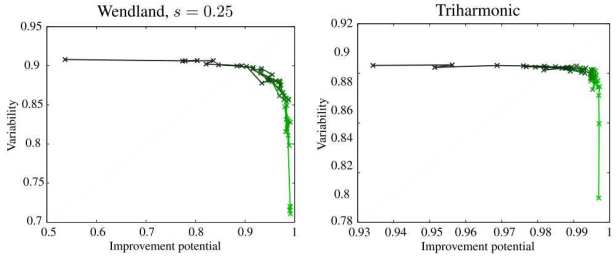

Fig. 6. Multi-objective optimization of potential RBF kernel set-ups under the principle of evolvability, taken from [30]. Left: RBF based on Wendland kernels [31], Right: RBF based on triharmonic kernels [23].



Fig. 7. Left: Uniform setup with high variability. Right: Adapted set-up with high improvement potential, taken from [30].

Based on the optimization it is possible to calculate RBF set-ups in an automated way avoiding a manual set-up process. The results balance between set-ups focusing exploration (high variability) and exploitation (high improvement potential), see Fig. 7. Note that each star in Fig. 6 marks the result of one optimization run with different RBF kernel distributions for matching the shape in Fig. 5. Practically speaking, these plots are very helpful for a designer/engineer to select the optimal deformation set-up under her/his assumptions on the potential exploration/exploitation character of the optimization. If exploration is expected, one would choose a high variability set-up. If exploitation is expected, one would choose a high improvement potential set-up. However, from visual inspection of the plots, set-ups around the knee point are most recommended since they comprise an optimal trade-off between exploration and exploitation and would cover eventual uncertainties and noise in the follow-up design optimization. So, choosing a set-up with a slight drop in improvement potential offers a huge increase in variability resulting in a substantially higher evolvability.

On a practitioner's side note, we have additionally shown that a tremendous gain for design optimizations using linear deformations is achieved by the orthogonalization of the deformation matrix U. Orthogonalization guarantees that the above-mentioned regularity R, expressed as condition number of U, equals 1 resulting in much faster convergence [32].

IV. Evolvable Prototyping: Platform Optimization

Platforms and platform systems are commonly exploited during industrial product development and are especially powerful for complex engineering systems. Platforms in the sense of a hardware prototype found the basis to develop and manufacture products in a cost-efficient way. A platform has to be highly evolvable to allow the dynamic integration of new or updated product specifications to avoid a complete redesign. In the following, our insights on a universal electric motor (UEM) product family optimization are given targeting a Pareto front of trade-off solutions for high performance platforms. An important aspect is the self-organization of parameter commonality in product family optimization.

Developing methods for the design of a compact product family implements the goal of optimal combinations and tweaks of modules applied to one or more base platform(s). A (hardware) platform is determined by the degree of commonality between the products, which are derived from this platform. Of course, while product variants need to be fast and flexibly adjusted to market needs, from manufacturing and development point of view an underlying product platform with a large number of common parts is required to increase cost efficiency. With respect to commonality two extremes exist which span the search space with minimum cost and low product performance on the one end, and maximum cost and global optimality (customization) of each product on the other end. In platform optimization these are called *total platform* where all design parameters are the same over all product variants and *null platform* where each product variant comprises its own optimal parameter settings.

In [33], an evolutionary optimization has been carried out to find the Pareto optimal platforms lying inside the search space described above for the unbiased universal electric motor (UEM) problem [34]. The implemented optimization framework based on NSGA-II and a computationally efficient concatenation of Pareto fronts solves the UEM problem without the need to seed the parent population with expert solutions as proposed in [35]. The framework consists of two cascading loops. Over the iterations of the outer loop the optimization algorithm evolves the platform parameters, i.e. it determines which parameters of the parameter set are common among all products and their values. The inner loop utilizes the current platform parameters of the outer loop and optimizes the remaining parameters for 10 different engine variants.

Conceptually, the outer loop takes care of the evolutionary self-organization process of the platform parameters utilizing the platform adaptability to the final engine specifications. Each platform performance for several product specs is measured by the inner optimization loops. Both processes

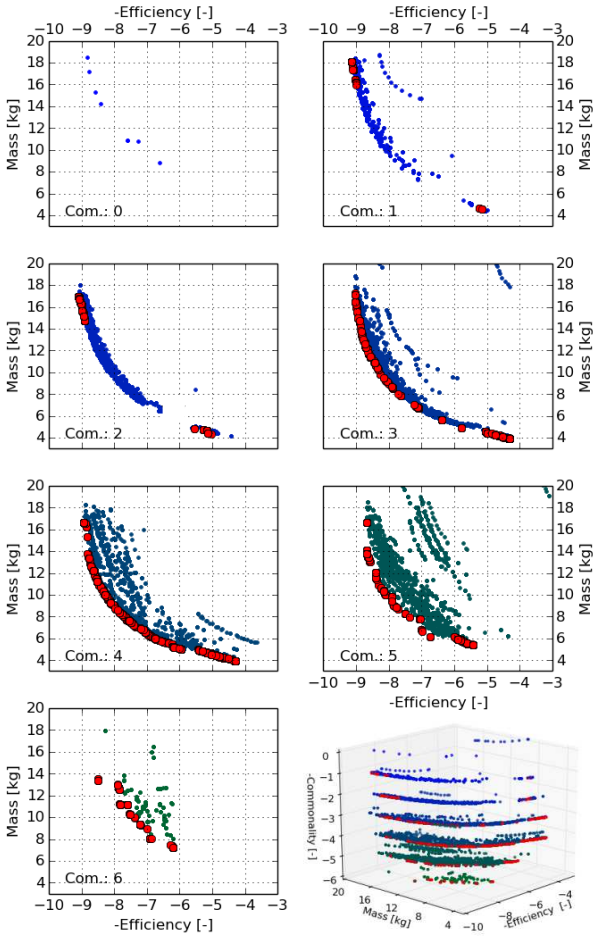

Fig. 8. Optimization results. Plots show the cross section results of the 3D Pareto front for platform commonalities Com.: 0 to Com.: 6, taken from [33].

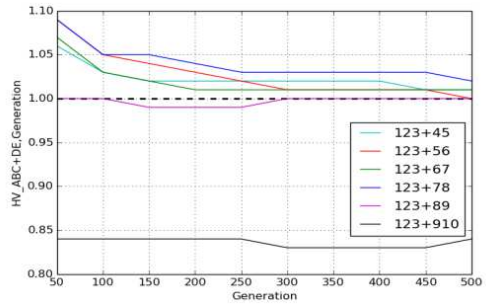

Fig. 9. Comparison of set-ups 123+45 to 123+910, taken from [33].

optimize for the set of evolvable platforms. Because of the stochastic population-driven approach, the evolutionary design optimization results in a three-dimension Pareto front for engine efficiency, sum of mass and commonality allowing an *a posteriori* specification of common parameters and their values. The complete optimization results in about 40000 product family solutions shown in Fig. 8 where each colored dot marks one Pareto dominant solution, which is based on an evolvable platform. Each Pareto solution offers the designer/ engineer to choose from a different number and combination of common (platform) parameters with different values.

To evaluate the evolvable character of product platforms, in a second scenario the influence of a dynamic change in the product specifications has been studied in the platform family optimization. For each specification change, two optimizations have been conducted and their performances have been evaluated by comparing the quotient of the normalized S-Metric-Hypervolumes [36]. First, based on a platform family which has been optimized solely for 3 product variants (denoted as P123), a follow-up optimization has been carried out adding 2 additional product specifications (denoted as P123+DE)[1]. Second, the cascaded evolutionary optimization has been started from scratch for the different product variants. Fig. 9 provides the performance results of the evolutionary optimizations. The graphs visualize the speed advantage, i.e. values larger than 1.0 when using an evolvable platform family, over the course of generations. Intuitively, the pre-initialization gives an advantage of up to 9 % for small product differences, which vanishes with increasing product differences. Since in real-world applications the differences over time are mostly unknown, a good strategy is to combine good solutions achieved in former optimization runs with randomly initialized solutions in the starting population as typically done in dynamic evolutionary optimization to cover a broader range of uncertainty.

V. Conclusion

By the constant rise of simulation and modeling capabilities, the trend is towards the optimization of holistic systems, which allows the function evaluation of virtual prototypes towards potential customer expectations and robust specifications to capture a broad spectrum of uncertainties. Allowing a system to be both, complicated and complex, engineering systems naturally require a constant improvement in both areas of complicated systems and complex systems. In the present paper, evolvability is focused since it is identified as a major actor to improve the handling of complex systems in dynamic environments and allows an emergence of system preparation for the unknown.

Following present practically relevant research, this paper shows different ways from a practitioner's point of view to improve the evolvability of a system. Key qualities of evolvability allow a comparison and better understanding of representations in general, the derivation of quantitative measures for evolvability allows an automated online adaptation of representations, and evolutionary optimization supports the process of the self-organization of platform commonality well. Nevertheless, for complex systems it is important to remember that an optimization result found at a specified point of time will not finally solve the problem in a changing environment but directly is a starting point for a follow-up adaptation process towards new frontiers.

[1]The values D and E can be interpreted as the difference of the product specifications between P123 and D and E, i.e. D=4, E=5 would be more similar to P123 than D=9, E=10.

References

[1] INCOSE - International Council on Systems Engineering, "What Is Systems Engineering," retrieved from: https://www.incose.org/systems-engineering, 25. Jan. 2019.

[2] A. A. Minai, D. Braha, and Y. Bar-Yam, "Complex Engineered Systems: A New Paradigm," in Complex Engineered Systems, D. Braha, A. Minai, Y. Bar-Yam (Eds.), Springer, Berlin, 2006.

[3] R. Poli, "A Note on the Difference Between Complicated and Complex Social Systems," CADMUS, v. 2, n. 1, pp. 142-147, 2013.

[4] N. Hansen, and A. Ostermeier, "Completely Derandomized Self-Adaptation in Evolution Strategies," in Evolutionary Computation, 9(2), pp. 159-195, 2001.

[5] M. J. Berryman, and P. Campbell, "Some complex engineering principles," in Australia's Preeminent Systems Engineering and Test & Evaluation Conference, 2010.

[6] S. M. Göhler, D. D. Frey, and T. J. Howard, "A model-based approach to associate complexity and robustness in engineering systems," in Research in Engineering Systems, vol. 28, pp. 223-234, Springer, 2017.

[7] H. Lehmann, and S. Menzel, "Evolvability as the Concept for the Optimal Design of Free-Form Deformation Control Volumes," in Proceedings of IEEE Congress on Evolutionary Computation (CEC), pp. 1-8, Brisbane, 2012.

[8] B. Calcott, "Evolvability and Engineering," in Biology and Philosophy 29(3), 293-313, 2014.

[9] J. Cutcher-Gershenfeld, F. Field, R. Hall, R. Kirchain, D. Marks, K. Oye, and J. Sussman, "Sustainability as an Organizing Design Principle for Large-Scale Engineering Systems," in Proc. from MIT Engineering Systems External Symposium and Symposium Monograph, 2004.

[10] H. Fu, B. Sendhoff, K. Tang, and X. Yao, „Finding Robust Solutions to Dynamic Optimization Problems," in: Esparcia-Alcázar A.I. (eds) Applications of Evolutionary Computation, EvoApplications 2013, Lecture Notes in Computer Science, vol 7835, Springer, Berlin, Heidelberg, 2013.

[11] C. Y. Baldwin, and K. B. Clark, "Modularity in the Design of Complex Engineering Systems," in Complex Engineered Systems: Science Meets Technology, A. Minai, D. Braha, Y. Bar-Yam (Eds.), NY: Springer, 2006.

[12] A. Wagner, "Robustness and evolvability: a paradox resolved," in Proc. R. Soc. B, vol. 275, pp. 91-100, 2008.

[13] I. Paenke, J. Branke, and Y. Jin, "On the Influence of Phenotype Plasticity on Genotype Diversity," in IEEE Symposium on Foundations of Computational Intelligence (FOCI), IEEE Press, 2007.

[14] J. Lehman, and K. O. Stanley, “Improving evolvability through novelty search and self-adaptation,” in Proceedings of IEEE Congress on Evolutionary Computation (CEC), pp. 2693–2700, 2011.

[15] D. Sieger, S. Menzel, and M. Botsch, “On Shape Deformation Techniques for Simulation-based Design Optimization,” in New challenges in grid generation and adaptivity for scientific computing, SEMA SIMAI Springer Series, 2015.

[16] D. Sieger, S. Menzel, and M. Botsch, “RBF Morphing Techniques for Simulation-based Design Optimization,” in Engineering with Computers, vol. 30, n. 2, pp. 161-174, Springer-Verlag, 2014.

[17] T. W. Sederberg, and S. R. Parry, ”Free-form deformation of solid geometric models,” in Proc. of ACM SIGGRAPH, pp. 151–59, 1986.

[18] S. Coquillart, “Extended free-form deformation: A sculpturing tool for 3D geometric modeling,” in Proc. of ACM SIGGRAPH, 1990.

[19] W. M. Hsu, J. F. Hughes, and H. Kaufman, “Direct manipulation of free-form deformations,“ in Proc. of ACM SIGGRAPH, 1992.

[20] A. de Boer, M. van der Schoot, and H. Bijl, “Mesh deformation based on radial basis function interpolation,” in Computers & Structures 85, pp. 784–795, 2007.

[21] M. Botsch, and L. Kobbelt, “Real-time shape editing using radial basis functions,” in Computer Graphics Forum (Proc. Eurographics) 24(3), pp. 611–21, 2005.

[22] S. Jakobsson, and O. Amoignon, “Mesh deformation using radial basis functions for gradient-based aerodynamic shape optimization,” in Computers & Fluids 36(6), pp. 1119–1136, 2007.

[23] A. K. Michler, “Aircraft control surface deflection using RBF-based mesh deformation,” in International Journal for Numerical Methods in Engineering 88(10), pp. 986–1007, 2011.

[24] A. I. Heft, T. Indinger, and N.A. Adams, “Introduction of a new realistic generic car model for aerodynamic investigations,” in SAE 2012 World Congress, 2012.

[25] S. Menzel, M. Olhofer, and B. Sendhoff, “Direct manipulation of free form deformation in evolutionary design optimisation,” in Int. Conf. on Parallel Problem Solving From Nature (PPSN), pp. 352–361, 2006.

[26] G. E. Fasshauer, “Meshfree approximation methods with MATLAB,” in World Scientific Publishing, 2007.

[27] OpenFOAM: Open Source Field Operation and Manipulation, retrieved from: https://openfoam.org, 31. Jan. 2019.

[28] S. Menzel, “Evolvable Free-Form Deformation Control Volumes for Evolutionary Design Optimization,” in Proceedings of IEEE Congress on Evolutionary Computation (CEC), pp. 1388-1395, New Orleans, USA, 2011.

[29] A. Richter, J. Achenbach, S. Menzel, and M. Botsch, “Evolvability as a quality criterion for linear deformation representations in evolutionary optimization,” in Proceedings of IEEE Congress on Evolutionary Computation (CEC), pp. 901–910, 2016.

[30] A. Richter, “Evolvability-guided Optimization of Linear Deformation Setups for Evolutionary Design Optimization,“ Doctoral Dissertation, Bielefeld University, 2019.

[31] H. Wendland, ”Scattered data approximation,” Cambridge University Press, Cambridge, England, 2004.

[32] A. Richter, S. Dresselhaus, S. Menzel, and M. Botsch, ”Orthogonalization of linear representations for efficient evolutionary design optimization,” in Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1356-1363, Japan, 2018.

[33] T. Friedrich, and S. Menzel, “A Cascaded Evolutionary Multi-objective Optimization for Solving the Unbiased Universal Electric Motor Family Problem,” in Proceedings of IEEE Congress on Evolutionary Computation (CEC), Beijing, pp. 3184-3191, China, 2014.

[34] T. W. Simpson, Z. Siddique, and J. Jiao, “Product Platform and Product Family Design Methods and Applications,” Gardners Books, 2010.

[35] S. V. Alundi, T. W. Simpson, and P. M. Reed, “Multi-objective design optimization for product platform and product family design using genetic algorithms,” in ASME 2005 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Long Beach, USA, 2005.

[36] H. Ishibuchi, N. Tsukamoto, Y. Sakane, and Y. Nojima, “Indicator-based evolutionary algorithm with hypervolume approximation by achievement scalarizing functions,“ in Proc. of the 12th annual conf. on Genetic and evolutionary computation, pp. 527–534, 2010.