Chapter 16
Managing Design Change with Functional Blueprints

Jacob Beal, Aaron Adler, Fusun Yaman, Jeffrey Cleveland, Hala Mostafa, Annan Mozeika, Kyle Usbeck, Gretchen Markiewicz and Benjamin Axelrod

Abstract Long-lived complex electromechanical systems, such as vehicles or industrial machinery, often need to be adapted for new uses or new environments. Adapting the design for such a system is frequently complicated by the fact that they are often tightly integrated, such that any change will have consequences throughout the design, and must take many different aspects of the system into consideration. Functional blueprints simplify adaptation by incorporating the reasons for design decisions and their consequences directly into the specification of a system. This allows a human designer to be supported by automated reasoning that can identify potential conflicts, suggest design fixes, and propagate changes implicit in the choices of the designer. This chapter presents the functional blueprints approach in detail, including both review of prior work and new results.

16.1 Introduction

Complex electromechanical designs, such as robots, vehicles, and industrial machinery, tend to be "brittle," meaning that it is often difficult to modify any significant aspect of the design without triggering a cascade of complex, difficult to predict and often costly changes. Such cascading changes are the result of interlocking constraints between elements that are modified and other parts of the design. An expert engineer, in fact, would likely consider many of these consequential changes to actually be keeping the design the same. When considered from the viewpoint of knowledge representation, a contradiction of this sort, where complex changes are required to keep a property the same, often indicates a critical flaw in representation.

J. Beal (✉) · A. Adler · F. Yaman · J. Cleveland · H. Mostafa · K. Usbeck · G. Markiewicz
Raytheon BBN Technologies, Cambridge, MA, USA
e-mail: jakebeal@bbn.com

A. Mozeika · B. Axelrod
iRobot Corporation, Bedford, MA, USA

L. Redding and R. Roy (eds.), *Through-life Engineering Services*,
Decision Engineering, DOI 10.1007/978-3-319-12111-6_16

If system specifications were more closely aligned with the ways in which human experts conceive of and work with designs, then many consequential changes would be either implicit from the specification or simple to automate. This would facilitate the development of design automation systems capable of making similar judgements about how to best maintain design integration in the face of changes. For complex electromechanical systems, such as aerospace vehicles, such tools might be able to significantly decrease the time and cost of both initial development and through-life upgrades and servicing. At the lower end of the complexity scale, such tools could enable simpler systems such as tactical ground robots to be rapidly modified in the field by operational experts in response to their evolving needs. Functional blueprints [1] are a representation aimed at providing such adapt- ability, inspired by biological development.

Functional blueprints capture expert knowledge by specifying a design as a set of behavioural goals and topological constraints and a method for incrementally adjusting the design when those goals or constraints are not being met. This chapter introduces the concept of functional blueprints and their application to electromechanical design. In particular, the work presented here focuses on robotic design, where systems are typically complex and highly integrated, yet relatively small and inexpensive, using an example robot similar to the iRobot LANdroid, but simpler and less expensive, called the "miniDroid." Section 16.2 introduces the functional blueprint concept and how it can be applied to electromechanical design. Section 16.3 then examines how interacting networks of functional blueprints can be used to generate both parametric and qualitative adaptation of a design. Section 16.4 discusses extensions to these approaches that can enable greater design plasticity. Finally, Sect. 16.5 discusses currently feasible applications and presents key open problems.

16.1.1 Comparison with Alternative Approaches

Design automation for electrical and mechanical systems has a long history, in which many significant results have been attained (e.g., [2–4]). A number of evolutionary methods have also been developed (e.g., [5, 6]). Applications have been limited, however, primarily due to the complexity and lack of smoothness in the design spaces that must be searched. Commercial modeling and simulation tools have thus been generally restricted to parametric exploration of relatively simple subsystems (e.g., [7]). Constraint-based local search methods, such as Kangaroo [8] and Comet [9], attempt to address this problem by using local parameter changes to minimize constraint violations, similar to functional blueprints but without their encoded knowledge of local repair strategies.

Control theory also addresses problems of system integration, but generally has difficulty with large numbers of non-linearly interacting parts. A notable exception may be viability theory [10], but its applicability is limited as it requires a system to be specified completed in terms of differential equations.

Other approaches to adaptive design of functional structures includes work on distributed adaptive construction [11–13], and various projects in self-reconfigurable robotics e.g., [14–16]. These approaches, however, are generally intended for more homogeneous and loosely coupled systems and would be difficult to adapt to electromechanical design. Recent work on "morphogenetic engineering" as proposed by Doursat [17] and for robotics in [16, 18], aims to support more heterogeneous systems. In particular, Doursat has laid out a framework for evolvable pattern formation [19], while Meng et al.'s generate patterns coordinating the configuration of a modular robot [20]. A more formal mathematical model can be found in [21], though the representational consequences are not yet explored.

16.2 Functional Blueprints

Functional blueprints, as defined in [1], specify a design in terms of behavioral goals and a method for adjusting the structure when those goals are not met, rather than a fixed structure. The approach is inspired by animal development, in which feedback processes maintain continuous integrated functionality across diverse subsystems such as muscles, nerves, and blood vessels as the animal grows from an embryo to a mature adult, despite the fact that the relationship and relative sizes of the elements making up these subsystems may change radically over time. Such decentralized adaptation is a critical enabler for the evolution of natural systems [22, 23].

Perhaps surprisingly, engineered systems also often show a family resemblance reminiscent of natural phylogeny, as in the iRobot product family shown in Fig. 16.1, to which also belongs the "miniDroid" robot (Fig. 16.3) used as a running example through the remainder of this chapter. This is due to the preference of human engineers to adapt functioning designs rather than to build from scratch. Functional blueprints aim to build on this parallel to enable engineered designs to exhibit the same power of facilitated change as is exhibited by natural systems.

Based on the feedback model of angiogenesis [24] and similar processes, a functional blueprint is thus defined as a collection of four elements: (1) a system behaviour that degrades gracefully across some range of viability, (2) a stress metric quantifying the degree and direction of stress on the system, (3) an incremental

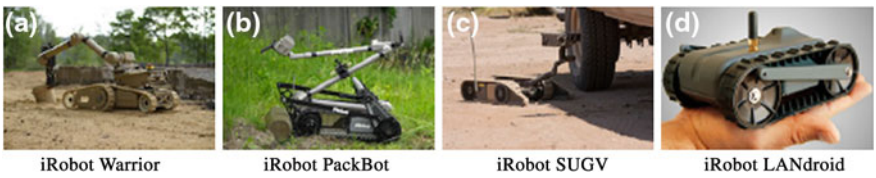

Fig. 16.1 Families of engineered systems often exhibit "phylogenetic" relationships similar to those of natural organisms. For example, these four iRobot products share a base body plan, including symmetric two-wheel treads, flippers coaxial with one wheel, and a top-mounted sensor/manipulator package. **a** iRobot Warrior, **b** iRobot PackBot, **c** iRobot SUGV, **d** iRobot LANdroid

program that relieves stress through growth, shrinking, or other structural change, and (4) a program to construct an initial viable minimal system.

In essence, this model uses stress as a coordinating signal by which independently developing subsystems are integrated. The stress metric and incremental program combine to shift the design back toward required functionality; graceful degradation ensures that there is a margin for error in the interactions between subsystems, and the minimal system ensures there is a viable place to start. A network of interacting functional blueprints may thus be viewed as a piecewise specification of a parametric model. Moreover, unlike a typical parametric model, this approach does not require on a closed-form relationship between parameters.

16.2.1 Application to Electromechanical Design

The Morphogenetically Assisted Design Variation (MADV) architecture shown in Fig. 16.2a applies the functional blueprint concept to the problem of electromechanical design adaptation [25]. Under this architecture, electromechanical designs are adapted following a three-phase loop: the current model is run through a set of functional blueprint evaluators to determine stress on each functional blueprint; from these stresses come requested adjustments of the design, which are blended together to produce an incrementally updated design. The loop continues to iterate until the design reaches a stable point—at zero stress if adaptation succeeds, and greater than zero if it runs into a contradiction it cannot resolve.

Users control design variation either directly by modifying parameters, or indirectly through modifications of the environment for simulation-based evaluations [e.g., changing the height of an obstacle that a robot is expected to climb over through a user interface like that in Fig. 16.2b]. In either case, this modification injects stress into the system, which causes the values to begin adjusting toward a new equilibrium. The user of the architecture can then observe the ongoing process

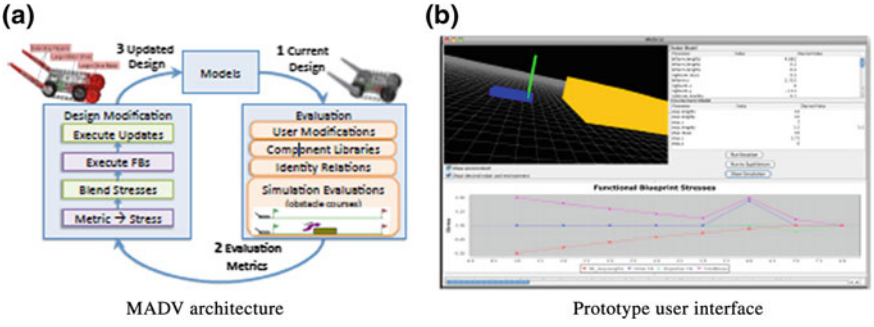

Fig. 16.2 **a** MADV architecture: designs are adapted following a three-phase loop: the current model is run through evaluators to determine stress on each functional blueprint; from these stresses come requested adjustments of the design, which are blended together to produce an incrementally updated design. **b** Screenshot of prototype MADV software

of adaptation to the new requirement, adjusting their specifications if they prove infeasible or to give hints to help the system if it gets stuck. When the user is satisfied, the final design can then be exported for further refinement or for fabrication.

For the work discussed here, the functional blueprints for electromechanical designs fall into four categories (presented in detail in [25]): (1) simulation-based blueprints measuring the ability of the systems as a whole to accomplish a task (e.g., a robot climbing a step), (2) families of COTS components (e.g., a collection of servo motors with various torque limits), (3) closed form relations (e.g., the identity relation between mass, density, and volume, the inequality between the vertical dimensions of a motor and of the robot body that contains it, or a functional specification requiring total mass less than a certain amount), and (4) user modification (e.g., a request to double the height of the step that can be climbed).

The collection of functional blueprints are integrated to form a complete network using a manifold-based representation that mixes geometric and topological elements [26, 27]. This allows the representation to include both architectural decisions (in the form of topological constraints and symmetries), geometric commitments (in the form of parametric values and geometry-based constraints), and functionality (in the form of functional blueprint constraints).

Finally, because parameters may have very different magnitudes, and because multiple functional blueprints may act on the same parameter, functional blueprints as implemented for MADV act on parameters only by expressing a stress in the range of [0,1] and a direction; the combination of stresses then produce value changes according to an adaptive process as described in Sect. 16.3.1.

Discussion and examples of applying the MADV architecture will largely focus on the miniDroid robot, shown in Fig. 16.3. This system was created by iRobot, based on the LANdroid from the robot family shown in Fig. 16.1 and slightly expanded and simplified to be a better target for investigation of adaptive design. Much of the work discussed in this chapter has been carried out using the miniDroid as a driving electromechanical design example, and it will thus be used as a running example. Figure 16.4 shows a functional blueprint network representing the miniDroid, including all design features at least 1 cm^3 in volume. In total, this comprises 23 components, 112 design parameters and 111 functional blueprints. Of the functional blueprints, three (step-climbing, self-righting, and fast driving) are

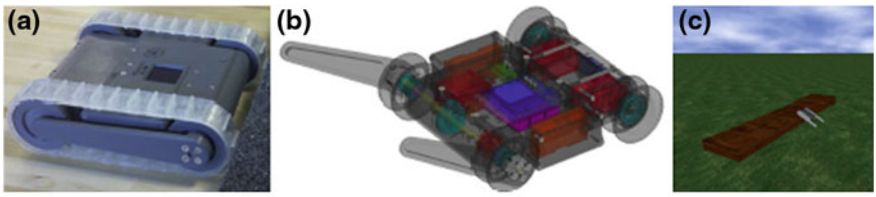

Fig. 16.3 miniDroid base robot design, in CAD (**b**) and reality (**a**). **c** Screenshots from the miniDroid ROS simulations

Fig. 16.4 Parameters, constraints, and initial values for the miniDroid: *blue* is parameters with an explicit initial value, *purple* is parameters whose value is inferred, *green* indicates constraint relations, and arrows link constraints to the parameters they affect (Color figure online)

evaluated with simulations implemented using ROS (Robot Operating System) [28] and the Gazebo simulator, while the remainder are either closed-form or represent COTS components.

16.2.2 Diagnosis and Assistive Design

Functional blueprint representations of electromechanical design can also provide useful services at lower levels of automation. In particular, since functional blueprints capture the intentions and requirements of a design, they can be used to assist a human designer with diagnosis of problems and suggestions for their solutions. This is a much less radical change to existing processes than automatic adaptation, yet still addresses many of the key challenges that motivate the investigation of automatic adaptation.

Figure 16.5 shows an example of a transcript from a prototype functional blueprint diagnosis system. The text for the explanations is generated automatically from the structural relations encoded in the functional blueprints and the parameters

```
Assuming fixed value for Instrument Package Mass and Instrument Package Volume

Detected 2 problems in current design (in descending order of importance):
Self Righting: Bot Body Length and Bot Body Mass are too large, relative to Flipper Motor Torque
Layout Geometry: Instrument Package Volume is too large, at 0.0100 Meters^3,
  Should be significantly smaller than Bot Body Volume, which is 9.43e-4 Meters^3

Suggestions to fix design problems:
  Decrease Bot Body Length from its current value of 0.200 Meters
  Increase Flipper Motor Torque from its current value of 1.22 Newton-Meters
  Increase Bot Body Volume from its current value of 9.43e-4 Meters^3
```

Fig. 16.5 Example transcript generated by electromechanical diagnostic system based on functional blueprint representation

they interact with. These explanations may be able to clarify the relations of a design to a human, and could form the basis of "expert advisor" systems for assisting domain experts in adapting designs.

16.3 Adapting Electromechanical Designs

Having captured the relationship between form and function as a network of functional blueprints, integrated adaptation of an electromechanical design may then be carried out as an iterative process of incremental adjustments. Many adaptations can be effectively carried out entirely by parametric variation, adjusting effective step size to optimize convergence speed while maintaining coherent design. Larger scale changes in specification, however, may also require qualitative changes in the collection of components making up the design.

16.3.1 Convergence of Functional Blueprint Networks

Functional blueprints deliberately do not contain any specification of how they are expected to interact with other functional blueprints: to do so would require an exponential number of relationships to be considered and would prevent them from being modular and capable to being reused in new designs. Instead, functional blueprints interact indirectly, by changing parameters such that stress is induced in other functional blueprints. That stress then induces those other functional blueprints to act, which may induce other stress, etc., propagating changes through the design. This raises a critical question, however: under what conditions is it possible to ensure that the stress thus generated will converge and return to zero (meaning that an acceptable adaptation has been found) in a reasonable amount of time?

The graceful degradation property of functional blueprints ensures that it is always possible to maintain the integration of a design. As proved in [1], it is always possible to reduce step-size such that no parameter ever saturates on stress. This does not, however, guarantee progress toward a user's desired specifications. Consider, for example, if one simultaneously requires a miniDroid to climb twice as tall a step and to be small enough to fit in a pocket: this may simply be physically impossible. It is also possible that a path to a solution may exist, but that interactions in the functional blueprint specifications may render it inaccessible to any particular algorithm.

If all functional blueprints are linear in their interactions, then convergence can be guaranteed, as demonstrated in [25]. Furthermore, the speed of convergence is quite rapid, outperforming genetic algorithms by more than two orders of magnitude. In most electromechanical designs, however, many interactions are non-linear; consider for example, that scale typically has a quadratic relation with strength and cubic with mass. Navigation of the stress space may then be treated as a non-convex

optimization problem, around which there is already well-established literature. The graceful degradation property of functional blueprints is helpful, as it ensures a relatively smooth stress function with response to any given parameter, and thus smoothness in the joint space as well. Unfortunately, many non-convex methods are still extremely computationally expensive, particularly given simulation-driven functional blueprints that lack a closed-form expression for the relationship between parameters.

Even simple heuristic methods, however, appear to be able to support large-scale variation in a real electromechanical design such as the miniDroid. One such heuristic that has been investigated for balancing convergence speed versus stability is to adaptively modify the size of each incremental adjustment of a parameter based on global and local stress. Specifically, the heuristic for parameter adjustment is:

$$S_p = \max_r \{s_{r,p}\} \cdot \frac{\sum_r s_{r,p}}{\sum_r |s_{r,p}|} \quad S_M = \max_p \{|S_p|\} \quad \Delta_p = \varepsilon \frac{S_p}{S_M} \tag{16.1}$$

where the $s_{r,p}$ is the stress exerted by relation (functional blueprint) r on parameter p and ε is the current incremental scaling factor for step size. The relative stress on a parameter S_p is then proportioned based on the degree of conflict between the stresses exerted on it by relations in the network, and the step size Δ_p proportioned based on the maximum stress S_M in the network and the incremental factor e. Finally, each parameter is changed by multiplying its current value by $1 + \Delta_p$. The size of e is critical to convergence: the larger that e is, the faster that the system will converge, but if it is too large, then parameter values will oscillate and possibly become unstable. The critical value for e may be difficult or impossible to determine statically (and indeed, may vary depending on parameter values). It is therefore useful to adaptively select the value of e based on the observed behaviour of the system. One simple heuristic which performs well is to examine the value of stress over a k-sample window: if stress is steadily decreasing, then multiply e by 2; if more than some threshold amount of oscillation is observed, then divide e by 2. Figure 16.6 shows examples of stress rising and converging during large-scale design adaptation using these heuristics (as well as qualitative design change at local minima, as discussed in the next section). Note in particular the transitions between oscillatory and smooth behaviour, as e is adjusted to attempt to control the rate of descent.

Using lightweight heuristics such as these, the time to execute one iteration of incremental adaptation is driven primarily by the cost of evaluating the functional blueprints, particularly those that are evaluated by simulation. Here, it is possible to greatly accelerate adaptation by exploiting the graceful degradation property of functional blueprints. Typically, only a small fraction of the relations in a system are simulation-driven functional blueprints; most others implement fast-evaluating relations such as specifications (e.g., a system mass limit), identity relations (e.g., total mass being the sum of component mass), physical relations (e.g., mass equals volume times density), or availability of components (e.g., dimensions vs. torque

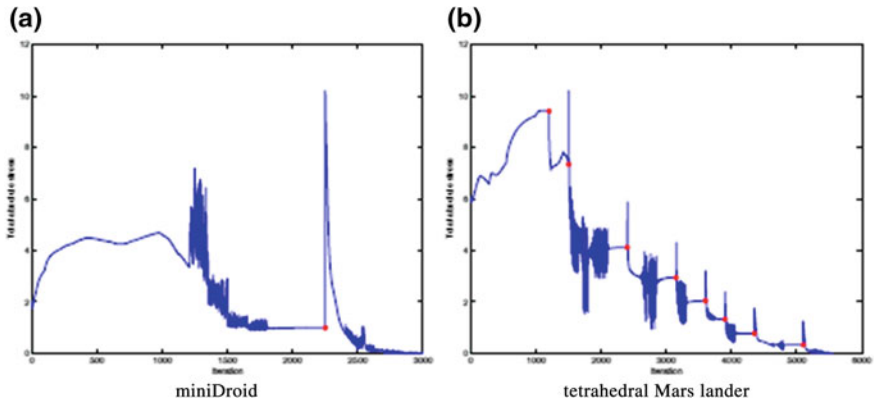

Fig. 16.6 Example traces of stress over time for constraint resolution including functional blueprints for qualitative change. Topological changes are marked with a *red* star; note that stress resolution reaches a stable "stalled" point before each change. **a** Shows a miniDroid adapting as the step size is raised from 10 to 25 cm, including shifting from one to two flipper drive motors to maintain self-righting with increasing mass. **b** Shows a tetrahedral lander adding motors until ultimately there are nine motors per panel as instrument package mass and power demand both increase 10-fold

for a family of COTS servo motors). Rather than compute simulations with every iteration, the stress value of each simulation may be cached and reused, such that there is only one simulation every n iterations. If the simulation has the required graceful degradation property, then its stress should not change significantly in a short time, so the rest of the network can continue adjusting while reducing the number of simulations required (and thus the approximate time) by a factor of n. Experiments have shown that it is likely possible to obtain one to two orders of magnitude speed-up with this mechanism, depending on the particulars of the design.

16.3.2 Parametric Versus Qualitative Design Changes

The discussion thus far has focused on parametric design changes, in which the parameter values of a blueprint network are adjusted, but the network (i.e., the set of components and their key topological relations) remains unchanged. Qualitative changes in design, on the other hand, make changes in the network of interacting parameters. For example, a new component may be added, an existing component removed, or the geometric relations of components significantly changed. Typical reasons for a human designer to make such a change include adding a new type of function, limitations in available components, and indirect geometric interactions of components.

Work thus far with functional blueprints has primarily focused on qualitative change in response to component limitations. Several approaches to indirect geometric interactions will be discussed in the next section, while decisions about functional goals may be properly left to humans. Topological change in response to component limitations is based around the addition of a "component" functional blueprint that associates together the set of parameters describing a quantized component with the set of relations bounding the component. Such functional blueprints are triggered only when the system reaches a stress minima, at which point the most stressed component in the network adjusts the number of instances of the component, splitting or merging component instances as indicated by the blueprint's prescription for the current stress. When a single component splits into multiple components, the functional blueprint network changes, adding "set" parameters for those properties that scale with number (e.g., the mass and torque of motors, but not their maximum rotational speed) and rewiring other relations to connect to the set or original parameters depending on the class. A complementary change occurs when a component set merges into a single component.

Both quantitative and qualitative variation have been tested by specifying large changes to the values of functional parameters of the miniDroid (e.g., the step height to be climbed, the amount of undercarriage clearance), and demonstrating that the network of functional blueprints successfully adapts, returning to a zero-stress state. Figure 16.6a shows an example of a stress trace from a typical such adaptation experiment, in which changing the step height from 10 to 25 cm causes parameter values to change throughout much of the design, as well as causing a change from one to two flipper drive motors in order to maintain self-righting with increasing mass. Notice the local minimum just before the number of flipper drive motors increases, and the rapidly dispersed pulse of stress afterwards, as the new values propagate through the network.

16.3.3 Reusability of Functional Blueprints

It is important to ensure that functional blueprints can often be reused from one design to another design, because it allows the effort required to create a functional blueprint to be amortized across the benefit of its use in many systems—a reuse expected to be further facilitated by the adaptivity inherent to the functional blueprint concept. This has been tested by constructing a design that reuses the self-righting functionality of the miniDroid, but is generally extremely different in form: a simplified tetrahedral Mars Lander based off of the landers for NASA's Spirit and Opportunity rovers.

These landers used a tetrahedral shape to ensure that the rover they delivered would be upright: three of the four panels open away from the fourth, like petals of a flower, so that if the rover lands on any face other than the intended bottom, it will be flipped upright by the opening of the panels. This functionality is similar to the miniDroid using its flippers to right itself, so the functional blueprints associated with this functionality should be reusable in the new design.

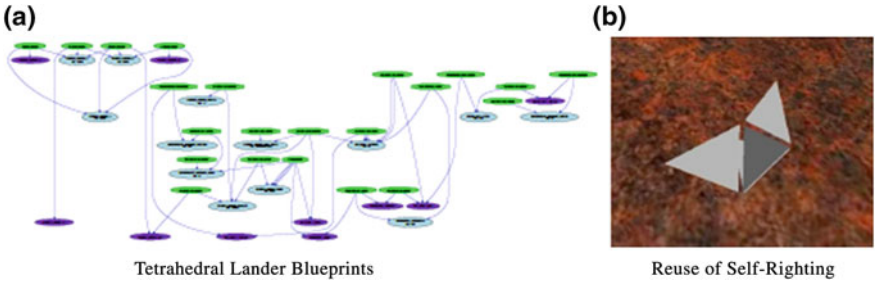

Fig. 16.7 Demonstration of functional blueprint reusability with a simplified design for a tetrahedral Mars lander based off of the landers for NASA's Spirit and Opportunity rovers: **a** shows the functional blueprint network for the lander, **b** shows a screenshot from a simulation carried out for the self-righting functional blueprint, one of a number reused from the miniDroid

Figure 16.7a shows the functional blueprint network for a simplified tetrahedral lander containing an instrument package attached to its base and solar cells on the interiors of the "side" panels, which are exposed when the tetrahedron opens, containing a total of 11 modules (4 panels, 3 solar panels, 3 flipper motors, and 1 instrument package). The symmetries of the design allow these component to be described with only 23 design parameters (abstracting away details of the instrument package contents), which are connected together by 22 constraints. Of these constraints, 18 are reused, including the flipping functional blueprint and the motor library. In many cases, reuse entailed minor modification of parameters [e.g., changing to Mars gravity, linking the flipping blueprint with the alternate simulation shown in Fig. 16.7b], but the main work required for each functional blueprint design was reused. As a result, creation of the tetrahedral lander's initial design and blueprint constraint network took only 2 h.

This design, like the miniDroid, enables large-scale design variation to be driven by changing critical parameters and allowing the rest of the parameters to adjust accordingly. For example, Fig. 16.6(b) shows the result of increasing the mass and power demand of the instrument package 10-fold. The lander grows in size, increases its solar panels to deal with the expected higher demands from a larger package, and undergoes qualitative change in the number of motors per panel, eventually finding that nine motors per panel are required to ensure that the lander can right itself. These appear are fairly regular intervals as the system gracefully moves towards resolution, stalling just before the introduction of each motor.

16.4 Expanding the Plasticity of Design

The adaptations considered thus far have been limited to relatively simple geometric forms and relations. The mixed geometric-topological representation used by MADV, however, is capable of supporting a much broader variety of complex

geometric forms [26, 27]. This potential for flexibility is both an advantage, in that it can in principle support plastic deformation of designs, and a challenge, in that the number of such possible deformations is extremely large.

Once again, the biological roots of functional blueprints provide inspiration for how to tackle this problem, though the answers are less well developed than for the more constrained forms of design change discussed in the previous section. In animals, the process of morphogenesis development of the organism's basic form from an initial egg effectively specifies a hierarchy of relationships and coordinate systems that "canalize" which changes of form are simple and which are complicated [22, 29]. As development continues, function-based feedback drives many forms of system integration [23], such as the ramification of blood vessels (driven ultimate by oxygen demand from tissue and by tension in the walls of blood vessels) and the assortation of neurons to control muscles (driven by competition for effective control).

Both of these forces are known to greatly facilitate the resilience and evolution of biological organisms [22, 23], so such developmental processes may also provide a useful model for facilitating greater plasticity in electromechanical designs. Such developmental models are in effect applying functional blueprints on a second and finer level, not just in transitioning from design to design, but also as a feedback process in the translation of a design specification to a geometric form that can then be evaluated by the "higher-level" functional blueprints already considered.

References [26, 30] present a variety of approaches for integrating such developmental models with the MADV architecture. At the coarsest scale [30] presents a "tissue development" model in which the topological and parameter relations of a design are generated by applying a set of developmental rules. Each developmental rule is an independent and asynchronous operation that applies a sequence of operators (the rule's body) on any tissue that matches the rule's preconditions. Executing a set of rules on an initially undifferentiated electromechanical body ("egg"), results in a body plan of a design "fetus" made up of various "organs" like limbs, flippers and wheels—the topological component of the mixed topological/geometric representation. Parameters in the rules become the geometric parameters constraining the topological elements generated by applying the rules, with design symmetries emerging from the repeated application of a single rule. Figure 16.8a shows an example of the basic miniDroid body-plan generated by a set of 12 rules operating in eight conceptual stages. Detailed description of the rules and stages may be found in [30].

The layout of wires, cables, and other such connections is another area of design where plasticity of form is useful, as these often have a great deal of flexibility in how they can be routed from place to place within a design. Making analogy to the chemotactic process by which neurons grow axons to their appropriate targets in biological development, routing of wires and other linear connectors can be accomplished by "seeding" each connector at one end-point and simulating an informational gradient from the other [26]. The connectors are then routed by climbing up the gradient, avoiding one another and fixed elements of the design as they grow.

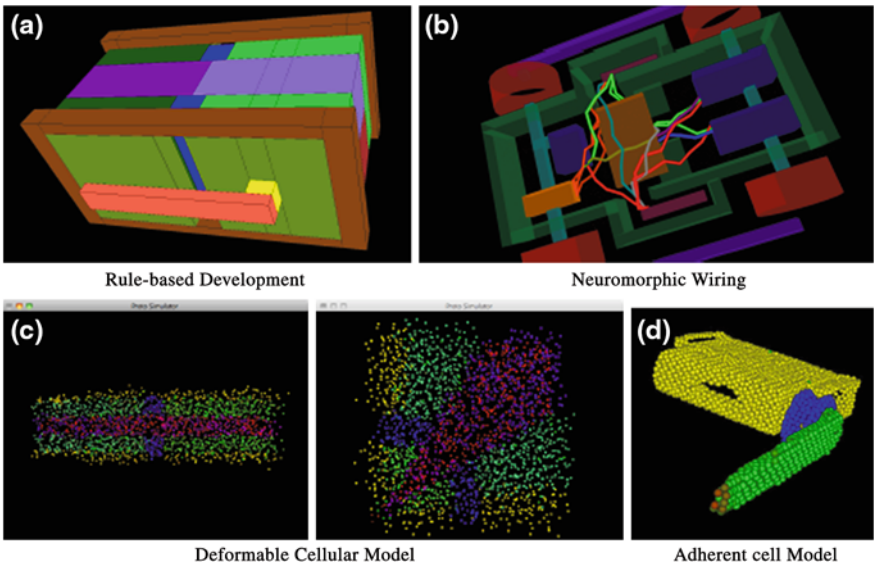

Fig. 16.8 **a** Rule-based development of a miniDroid topology from an initial undifferentiated "egg." **b** Neuromorphic routing connecting electronic components in the interior of a miniDroid design with wires carrying power (*red*), ground (*green*), and signal (*other colors*). **c** Distortion tolerant developmental program using the Proto manifold model to automatically adapt to execution on a modified underlying shape, e.g., shifting from a thin *rectangle* (*left*) to an *square* with a twisted coordinate system (*right*). **d** miniDroid flipper growing from the tip (*green*) in adherent cell simulation using MASON (Color figure online)

Figure 16.8b shows an example of this wiring model, from [26], being applied to lay out power, ground, and signal wires connecting the batteries, CPU, encoder, and motors of a miniDroid.

At a finer grain, Figs. 16.8c, 16.8d show cellular models of development [26]. These cellular models have the advantage that the distribution of "cells" can directly represent arbitrarily complex shapes, but the disadvantage that it is necessary to interpret these shapes to re-connect them with the parametric relationships that higher-level functional blueprints act upon. Another challenge is the tension between the resolution of the cells and the cost of simulation. Figure 16.8c shows how a fine-grained cellular model, implemented using the manifold geometry operations of the Proto [31, 32] aggregate programming language, allows a developmental plan to distort to match the conditions of its execution. In this case, the layout of electronics, wheels, and motors in a miniDroid "body plan," executing on a model of 2,000 cells distributed through a rectilinear 3D volume of space, is changed in proportion (left) and twisted by a change in coordinate system (right), showing the inherent geometric adaptation of the program. Further coherence in adaptation can be enabled by soft-body simulation, which can allow design elements to adhere, compress, deform, penetrate, or otherwise physically interact with one another for co-adaptation during the developmental process. Figure 16.8d

shows an example of a computationally inexpensive soft-body model implemented using MASON, a Javabased multi-agent simulator [33], implementing tapering of a miniDroid flipper by means of an adhesion-based growth process.

16.5 Applications and Open Problems

This chapter has presented the concept of functional blueprints, showing that they are a viable and potentially scalable approach to adaptation of complex electromechanical designs. Applied in combination with unified topological-geometrical representation and self-tuning step sizes, functional blueprints can adapt complex designs effectively across a large range of variation, producing both quantitative and qualitative changes that maintain functionality in a changing electromechanical design. Functional blueprints are also composable and reusable, as demonstrated by the transfer of blueprints from the miniDroid design to the tetrahedral lander design, and the approach may be further extended to allow greater adaptability through the plastic deformation of components.

The visionary goals for this approach are two-fold. The first is to allow non-experts to produce design variations that satisfy new requirements, even without a good understanding of subsystems, simply by indicating the critical changes that are needed and allowing the rest of the complementary changes to propagate automatically. The second is to enable a continuous design and manufacturing cycle, in which emerging additive manufacturing technology joins with a functional blueprint driven radically decreased cost of redesign to enable even highly complex electromechanical systems like aerospace vehicles to be updated on the fly, rapidly incorporating new technologies and responding to changing requirements. Realizing these visions will require an investment to develop libraries that associate existing CAD components with corresponding functional blueprints, as well as development of appropriate user interfaces, and additional work to enhance the speed and reliability with which large networks of functional blueprints can be guaranteed to converge.

Because functional blueprints encode knowledge about a design, they can also be used to assist human designers in other ways than adaptation. In particular, functional blueprints can not only detect potential design problems but can be used to generate human-readable explanations of the causes of those problems and suggestions for approaches to fixing them. This holds the potential for improvements in the way that mechanical engineering is conducted and taught.

In the practice of mechanical engineering, functional blueprints can be used to import the software notion of continuous integration into electromechanical design. Continuous integration is an important tool for rapid and reliable software engineering, in which every incremental step in the realization of a design is automatically tested against a suite of “regression tests” ensuring that important existing functionality is not endangered by progress in other areas. By evaluating the ability of a design to satisfy requirements, functional blueprints could be used to

effectively implement such regression testing, decreasing the cost of integration and the number of design problems identified after production.

Finally, functional blueprints could also be used in several ways to help educate students on electromechanical design and/or manufacturing. First, functional blueprints could be used as part of an active learning process to give a student instantaneous feedback on the strengths and weaknesses of their current design. Second, since functional blueprints can be used to adapt a design to find an integrated solution, this sort of "look-ahead" could be used either to give students hints to help with design or to evaluate what aspects of design a student is most struggling with, and therefore to adaptively present or recall relevant curriculum elements.

In summary: the engineering of complex electromechanical systems is an important problem that impacts society in myriad ways. Functional blueprints offer the potential to improve the engineering process at every phase: assistance in design, improved diagnosis of potential faults, simplification of through-life adaptation and redesign, democratization of engineering, and even improvement of the education of future electromechanical engineers.

References

1. Beal J (2011) Functional blueprints: an approach to modularity in grown systems. Swarm Intell 5(3):250–281
2. Campbell MI, Cagan J, Kotovsky K (1999) A-Design: an agent-based approach to conceptual design in a dynamic environment. Res Eng Design 11(3):172–192
3. Hoover SP, Rinderle JR (1989) A synthesis strategy for mechanical devices. Res Eng Design 1 (2):87–103
4. Fromherz MPJ, Bobrow DG, de Kleer J (2003) Model-based computing for design and control of reconfigurable systems. AI Magazine 24(4):120–130
5. Fan Z, Wang J, Goodman E (2005) Cutting edge robotics. In: Kordic V, Lazinica A and Merdan M (eds) Exploring open-ended design space of mechatronic systems, Pro Literatur Verlag, Germany, pp 707–726
6. Koza JR, Bennett III FH, Andre D, Keane MA (1998) Adaptive computing in design and manufacture, Springer, Berlin, pp 177–192
7. ANSYS, Inc. ANSYS DesignXplorer (2013) http://www.ansys.com/Products/Workflow+Technology/ANSYS+Workbench+Platform/ANSYS+DesignXplorer
8. Newton MAH, Pham DN, Sattar A, Maher MJ (2011) Kangaroo: an efficient constraint-based local search system. In: Principles and practice of constraint programming 17th international conference, CP, Italy, pp 645–659
9. Van Hentenryck P, Michel L (2005) Constraint-based local search, MIT Press
10. Aubin JP (1991) Viability theory, Birkhauser
11. Werfel J (2006) Anthills built to order: Automating construction with artificial swarms. Ph.D thesis, MIT
12. Werfel J, Nagpal R (2007) In: International conference on intelligent robots and system
13. Estevez N (2007) Functional blueprints: a dynamical systems approach to structure representation. Cornell University
14. Stoy K, Nagpal R (2004) Self-reconfiguration using directed growth. In: International symposium on distributed autonomous robotic system (DARS), Springer-Verlag, New York, pp 3–12

15. O'Grady R, Christensen AL, Dorigo M (2009) SWARMORPH: multi-robot morphogenesis using directional self-assembly. IEEE Trans Rob 25(3):738–743
16. O'Grady R, Christensen AL, Pinciroli C, Dorigo M (2010) Robots autonomously self-assemble into dedicated morphologies to solve different tasks. In: AAMAS, pp 1517–1518
17. Doursat R (2011) Morphogenetic engineering weds bio self-organization to human-designed systems. PerAda Magazine
18. Jin Y, Meng Y (2011) Morphogenetic robotics: an emerging new field in developmental robotics. IEEE Trans syst Man Cybern C Appl Rev 41(2):145–160
19. Doursat R (2008) Organic computing. In: Wurtz R (ed) Springer, Berlin, pp 167–200
20. Meng Y, Zhang Y, Jin Y (2010) A morphogenetic approach to self-reconfigurable modular robots using a hybrid hierarchical gene regulatory network. In: International conference on the synthesis and simulation of living systems, pp 765–772
21. MacLennan B (2010) Models and Mechanisms for Artificial Morphogenesis. In: Natural computing, Proceedings in information and communications technology, vol 2. pp 23–33
22. Carroll SB (2005) Endless forms most beautiful, W.W. Norton and Company, New York
23. Kirschner MW, Norton JC (2005) The plausibility of life: resolving darwin's dilemma, Yale University Press, New Haven
24. Carmeliet P (2003) Angiogenesis in health and disease. Nat Med 9(6):653–660
25. Adler A, Yaman F, Beal J, Cleveland J, Mostafa H, Mozeika A (2013) A morphogenetically assisted design variation tool. In: AAAI, pp 9–15
26. Beal J, Mostafa H, Axelrod B, Mozeika A, Adler A, Markiewicz G, Usbeck K (2012) A manifold operator representation for adaptive design. In: GECCO 2012, pp 529–536
27. Beal J, Adler A, Mostafa H (2013) Mixed geometric-topological representation for electromechanical design. In: Extended abstract at GECCO 2013, pp 105–106
28. Quigley M, Gerkey B, Conley K, Faust J, Foote T, Leibs J, Berger E, Wheeler R, Ng AY (2009) In: Proceedings of the open-source software workshop at the international conference on robotics and automation (ICRA)
29. Waddington CH (1942) Nature 150(3811):563
30. Beal J, Lowell J, Mozeika A, Usbeck K (2011) Using morphogenetic models to develop spatial structures. In: Spatial computing workshop 2011 at IEEE SASO '11, pp 85–90
31. Beal J, Bachrach J (2006) IEEE intelligent systems, pp 10–19
32. MIT Proto software available at http://mitproto.net. Retrieved 16 Sept 2012
33. Luke S, Cioffi-Revilla C, Panait L, Sullivan K, Balan G (2005) Simulation 82(7):517