Functional Blueprints: An Approach to Modularity in Grown Systems

Jacob Beal

Abstract The engineering of grown systems poses fundamentally different system integration challenges than ordinary engineering of static designs. On the one hand, a grown system must be capable of surviving not only in its final form, but at every intermediate stage, despite the fact that its subsystems may grow unevenly or be subject to different scaling laws. On the other hand, the ability to grow offers much greater potential for adaptation, either to changes in the environment or to internal stresses developed as the system grows. We may observe that the ability of subsystems to tolerate stress can be used to transform incremental adaptation into the dynamic discovery of viable growth trajectories for the system as a whole. Using this observation, we consider an engineering approach based on *functional blueprints*, under which a system is specified in terms of desired performance and means of incrementally correcting deficiencies. This approach is demonstrated by applying it to the integration of simplified models of tissue growth and vascularization, then further by showing how the composed system may itself be modulated for use as a component in a more complex design.

1 Introduction

One of the most remarkable facts about animals is that they are not generally injured by their own growth. An animal is composed of many tightly integrated systems, all interlocking in multiple ways. For example, bones fit together in joints that permit a useful range of motion, muscles attach to the bones in a pattern that allows them to work together effectively to move the body, the circulatory system delivers oxygen and nutrients to every portion of the bones and muscles via an intricate network of vessels, and their waste products are carried away for removal by the kidneys. As the

Jacob Beal
BBN Technologies, Cambridge, MA, USA e-mail: jakebeal@bbn.com

animal grows, from an embryo to a mature adult, all of these systems are constantly adapting in order to remain integrated and fully functional.

This is not generally the case for our current engineered systems. Many artifacts, such as cars and airplanes, have no real capacity for growth at all. In engineered systems that do grow, the growth is often accompanied by significant degradation of function as the existing balance of systems is disrupted and painstakingly reintegrated. Adding an extension to a house means months of dust, being unable to use existing rooms, and electrical and plumbing disruptions. Expanding the road networks of a growing city requires years of detours and traffic disruptions, not to mention economic disruption for businesses nearby the construction. Upgrading the software of a computer often requires a reboot and leaves a trail of incompatibilities and ongoing headaches. Beyond the obvious differences in mechanical and material properties, we simply do not know how to describe our designs in a way that allows for disruption-free growth. We may thus be led to consider languages for adaptable design, both to better understand animal development and also to improve engineered systems. This is particularly pressing given the rapid progress occurring in synthetic biology (e.g. engineered pattern formation [2] and standardized DNA assembly protocols [15]), where the systematic engineering of DNA programs promises to soon allow us to created engineered objects that are literally grown from living cells.

One particularly elegant example of growth and adaptivity in biological systems is the vascular system [6]. Under normal conditions, sufficient oxygen diffuses through the walls of capillaries into the surrounding tissue. When cells are not receiving enough oxygen, however, they become stressed and emit a chemical signal that causes nearby capillaries to leak. The vascular system also has an elegant program for regulating its capacity. When a capillary leaks often, a new capillary begins to grow out of the leaky area, increasing the available blood supply to the oxygen-starved region. Blood vessels are elastic, and when they are frequently stretched, the cells divide, increasing the capacity of the vessel; likewise, when frequently contracted, cells die and shrink the vessel. Thus, the vascular system incrementally grows and shrinks to match the demand of the tissues it serves, branching into underserved regions and adjusting the size of vessels to match the flow through them.

We may thus be led to consider an engineering approach of *functional blueprints* inspired by this and other similar adaptive biological systems. If each system is capable of operating under minor stress and of incrementally adjusting to decrease stress, then feedback between components should allow all the subsystems comprising a natural or engineered system to maintain a tight integration as the system grows, even if the relationship and relative sizes of subsystems are changing. Functional blueprints, as developed in [3], attempt to capture this by specifying a system in terms of desired performance and means of incrementally correcting deficiencies.

To understand the functional blueprint approach, we first consider how stress tolerance can enable integrated growth, then formalize this idea with a definition for a functional blueprint. We next demonstrate the functional blueprint approach by applying it to the integration of simplified models of cell density maintenance and vascularization producing synchronized tissue growth, then finally show how the composed system may itself be modulated for use as a component in a more complex design.

1.1 Related Approaches

Morphogenesis in natural systems has been a subject of intensive study. In recent years, deciphering of genetic mechanisms controlling development, such as how the *hox* gene complex produces the overall body plan of animals, has lead to a synthesis of evolution and development ("evo-devo") [7], and theories of how the adaptivity of organisms to body plan variations may facilitate evolution [10].

Inspiration from natural systems has led to investigation of how growable patterns might be programmed, generally focusing on the establishment of shape, with less attention to integration of function. Doursat, for example, has developed a *hox*-gene-based network model for artificial evolution of animal-like systems [9]. Similarly, the development of structure in growing plants has long been modeled at a high level by term-rewriting systems [14], which the MGS language extends into a general model of structure development through topological rewriting [16]. Other notable approaches include Coore's Growing Point Language [8], which uses a botanical metaphor to create topological structure and Nagpal's Origami Shape Language [13], which creates geometric forms through folding. Most similar to this work is Werfel's work on distributed construction, [17], which has been extended to the use of functional constraints to generate adaptive structure in response to environmental stimuli [18].

The problems of integration addressed in this paper are also related to control theory. Standard control theory, however, has difficulty addressing systems with large numbers of non-linearly interacting parts, which are typical of growing systems. A notable exception may be viability theory [1], a branch of mathematical theory which is intended to address such concerns.

2 Stress Tolerance Enables Integrated Growth

The basic insight enabling this new approach is as follows: *a stress-tolerant system can exploit its tolerance to navigate dynamically through the space of viable designs*. This is rather foreign to the typical engineering approach to failure tolerance. Usually, an engineer designing a system treats its ability to tolerate failures like guard rails on a highway: important for safety, changing terrible outcomes into merely bad, but never touched under normal circumstances. Alternately, though, we can treat the system's robustness as a guide, the way that a blind person might use a guard rail to follow the twists and turns of the road.

Under this alternate view, stress within the system becomes the *coordinating signal* by which independently developing subsystems are integrated. When the system

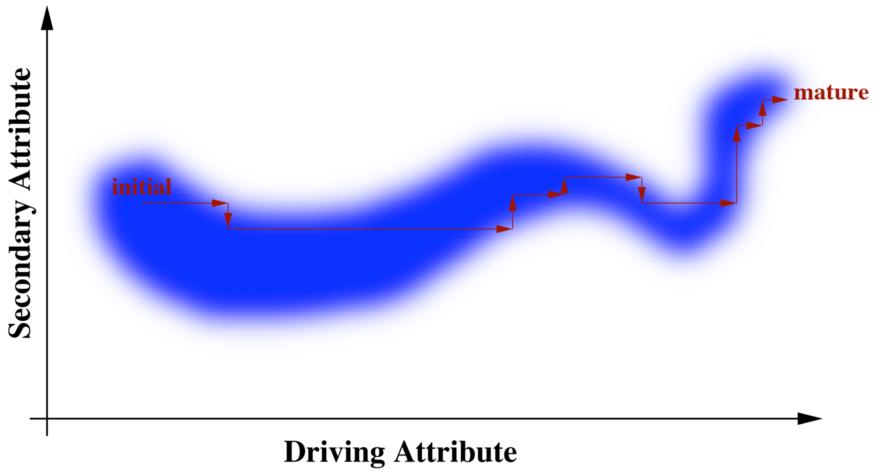


Fig. 1 In this abstract example, a growing system with two attributes uses stress tolerance to navigate through a complex viability envelope (blue). When unconstrained, the system grows its driving attribute (horizontal arrows). When the system's viability begins to be impaired (faint blue), it relieves that stress by adjusting its secondary attribute (vertical arrows). By repeatedly switching between driving growth and relieving stress, the system is able to navigate a complex viability envelope.

is far from the edge of its viability envelope, it can develop freely. When it comes near the edge, however, and its viability begins to be impaired, then the growth of the subsystems driving it to non-viability is slowed or stopped temporarily. Other subsystems, triggered to act by the increased stress, adjust to bring the system as a whole back within the viability envelope. The driving subsystems are then re-enabled, and the cycle of growth and correction begins again.

Critically, this is only possible if the system is able to determine the direction of stress, and if stress caused by one system can be relieved by adjustment of another. For example, if a beam's length has become wrong due to the change of structure around it, the beam will experience tensile stress if it is too short and compressive stress if it is too long. If only the magnitude of error is measured, then the beam cannot know whether it should grow or shrink to reduce stress, but if the direction of the stress is measured then the appropriate corrective measure becomes obvious.

For example, consider an abstract system with two attributes, whose combination is viable only in the complex envelope shown in Figure 1. The horizontal attribute drives system growth, increasing whenever the system is clearly viable (horizontal arrows). When the system's viability begins to be impaired (faint blue), the secondary attribute adjusts to correct (vertical arrows). Given some hysteresis in the switch between driving and correction, the switch in modes need only occur a finite number of times. By repeatedly switching between driving growth and relieving stress, a system may navigate a complex viability envelope.

Thus we see that it is possible to use systemic stress as a signal to coordinate the growth of independently developing subsystems. This will not, of course, work for all possible such viability spaces: if the viability space includes a "dead end" that the driving attribute can push into, then it cannot be successfully navigated without additional guidance. For many systems, however, such as those where the coordination problem is rooted in the difference of scaling laws—e.g., bone length (linear) vs. muscle cross-section (square) vs. lung capacity (cubic)—the viability space is guaranteed to be navigable. Note also that when stress is localized, the process of correction can be localized as well, allowing navigation to be parallelized when systems are not directly affecting one another. For example, different sets of developing muscles can be sore at the same time.

3 Functional Blueprints

Having made the observation that stress tolerance can allow a system to dynamically discover trajectories through its viability space, we can now take the next step and propose an engineering framework for predictably constructing such systems. Let us thus define a *functional blueprint* for some system X to consist of four elements:

1. A *system behavior* that degrades gracefully across some range of viability. Formally, if C_X is a manifold of possible configurations of system X, then it must be possible to establish a concave viability function v_X mapping $C_X \to [0,\infty)$ such that for any configuration c_X, only viable configurations have $v_X(c_X) > 0$,[1] and for any such configuration there exists a ball $B \subset C_X$ centered on c_X such that $v_X(B) > 0$.
2. A *stress metric* quantifying the degree and direction of stress on the system. Formally, let the stress metric $\mathbf{s}_X$ be a vector field on C_X such that $\mathbf{s}_X$ is the gradient of some legal viability function for C_X.
3. An *incremental program* that relieves stress through growth (or possibly shrinking). Formally, let this be a parametrized map $i_{X,\varepsilon,d} : C_X \to C_X$ that shifts a configuration by ε distance in the direction d.
4. A program to construct an initial *minimal system*. This initial minimal system, which we label X_0, must be viable, i.e., $v_X(X_0) > 0$.

Graceful degradation of system behavior asserts that the core functionality of the system must not have a sharp transition between viable and non-viable. The stress metric and incremental program combine to shift a degraded system's configuration back toward viability. Finally, the minimal system makes sure that there is some viable place to start.

To transfer these properties to a composite system, it is only necessary to ensure that the subsystems are coupled in such a way that the side effects of subsystems on

[1] Note that not all viable configurations need have $v_X(c_X) > 0$: the point is for the viability function to serve as a conservative guide for system growth, not to capture the precise boundary at which the system fails.

one another are incremental. Formally, the action of each subsystem X's incremental program on every other subsystem Y forms a continuous map, $\pi_{X,Y}$. Given such a coupling of functional blueprints, it is always the case that it is possible to adjust any given subsystem by some small increment without knocking any other subsystem out of its range of viability. This can be proved by construction:

Theorem 1. *Consider a system S, for which every subsystem has a functional blueprint, and let X and Y be subsystems of S. For any given configuration c_S, if $v_X(c_X) > 0$, then there exists a $\delta > 0$ such that $c'_S = i_{Y,\varepsilon,d}(c_S)$ has $v_X(c'_X) > 0$ for every d and $\varepsilon \leq \delta$.*

Proof. By graceful degradation, we know that there exists a ball B centered on c_X such that every point $b \in B$ also has $v_X(b) > 0$. By the continuity of the coupling map $\pi_{X,Y}$, we know that the preimage of $\pi^{-1}_{X,Y}(B)$ is an open set. Being an open set, the preimage must contain some ball B' of radius δ around the configuration c_Y. By the definition of an incremental program, any configuration c'_S accessible via subsystem Y's incremental program $i_{Y,\varepsilon,d}(c_S)$ is within the ball B' for $\varepsilon \leq \delta$. Since B' is a subset of $\pi^{-1}_{X,Y}(B)$, c'_X must be within B and therefore have $v_X(c'_X) > 0$. □

A simple growing composite system, such as the one illustrated in Figure 1, can thus be constructed simply by taking the composite stress to be the maximum stress of any subsystem and executing the incremental program of the maximally stressed subsystem. The system can then be navigated toward a desired mature form by driving any subsystem or collection of subsystems whenever the composite stress is low enough.

4 Example Application: Tissue Growth

Having proposed a framework for the design of grown systems, let us now demonstrate its feasibility by developing a simplified model of tissue growth, in which the growth of a sheet of cells is synchronized with the growth of the blood vessels that supply them with oxygen. This example system should not be regarded as a serious model of tissue growth, but as a cartoon to demonstrate the feasibility of the engineering approach under discussion.

This simplified model consists of two subsystems, each specified with a functional blueprint. The cell density subsystem attempts to keep cells packed at a moderate density via motion, reproduction, and apoptosis. A consequence of this density maintenance is tissue growth at an approximately constant rate of expansion: cells at the surface of the tissue generally have a low average density, since there are no cells to one side of them, so unless they are regulated otherwise, they will tend to reproduce. The vascularization subsystem, on the other hand, attempts to ensure that no cell is too distant from a network conveying oxygenated blood outward from a source.[2] These two systems are linked together by adding a regulatory input to the cell density subsystem, such that cells will not attempt to reproduce if they are oxygen-starved. The resulting composite system produces smoothly synchronized tissue growth, and can be modulated to produce shaped tissue by external regulation of either subsystem.

These models are developed and simulated using the Proto spatial computing language [4], a functional language that allows the programmer to specify aggregate behaviors using scalable geometric descriptions. Aggregate behavior descriptions are then compiled into a program to execute on each cell (every cell is given the same program) and an interaction protocol by which cells cooperate to approximate the desired aggregate behavior. Programs written in Proto are included in the present text for completeness, and to allow the interested reader to investigate them, if they wish, using the MIT Proto distribution [12]. Rather than go into a long digression on Proto, however, we will explain the meaning of the programs to the reader with embedded comments. Further details on Proto can be found in [4], [5], and the documentation accompanying the MIT Proto distribution [12].

4.1 Cell Density

In this simplified model, the base structure and expansion of a sheet of cells is produced by a system that attempts to keep cells packed at a moderate density. We can implement such a system as follows:

```
(def cell-density (grow shrink p_d)
  (let ((packing (num-nbrs)))
    (clone (and (and (< packing 8)  ; If too few neigbbors
                     grow)          ; and allowed to grow
                (< (rnd 0 1) p_d))) ; randomly reproduce
    (die (and (or (> packing 15)   ; If too many neighbors,
                  shrink)          ; or commanded to shrink
              (< (rnd 0 1) p_d)))) ; randomly suicide
  (disperse 9))  ; Move away from other nearby cells
```

Here the desired system behavior is to maintain a moderate spacing between cells, which exhibits graceful degradation if the cells have some tolerance for overcrowding or underpopulation. The system is thus stressed when there are too many neighbors (here defined as more than 15), too few neighbors (here defined as less than 8), or if the neighbors are not at a desired separation (here defined as communication radii of 0.6).

The incremental program relieves stress in a straightforward manner: when there are too many neighbors, the cell goes through apoptosis (cell death) with probability p_d, and when there are too few neighbors, the cell reproduces with probability p_d

[2] In this simplified system, venous return is not modeled, but could be implemented using a complementary mechanism.

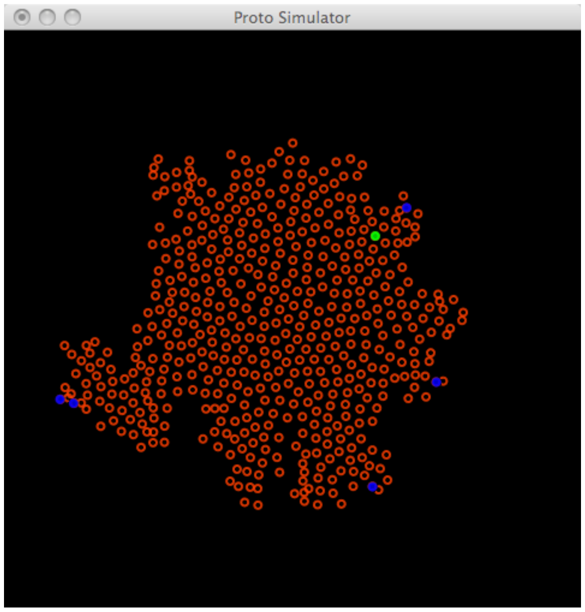


Fig. 2 Density maintenance and growth: simulation of an expanding sheet of cells, where blue cells are reproducing and green are dying.

(the `grow` enabling input and `shrink` forcing input allow these actions to be modulated by an enclosing system). When the neighbors are not at a desired separation, they move towards it using spring forces:

```
(def disperse (pfd-distance)
  (* (/ 1 (int-hood 1)) ; normalize over neighbors...
     (int-hood
      ;; the difference from preferred distance
      (let ((dr (- (nbr-range) pdf-distance)))
        ;; (multiplied by 1/10 when attracting)
        (* (mux (< dr 0) dr (* 0.1 dr))
           ;; turning distance into a vector to the neighbor
           (normalize (nbr-vec)))))))
```

in which attractive forces are weaker than repulsive forces such that the farther neighbors do not exert too much influence and collapse the diameter of communicating clusters.

Note that since cells at the surface of the sheet have an expected density half that of cells in the interior of the sheet, their density will be considered too low (except in temporary high-density pockets) and they will reproduce. This has the desirable consequence of continually expanding the sheet of cells such that the edge moves outward at an expected constant rate (Figure 2).

Figure 3 shows this cell density system in experiments where the growth parameter p_d ranges from 0.01 to 0.5. For each parameter value, 10 trials were run,

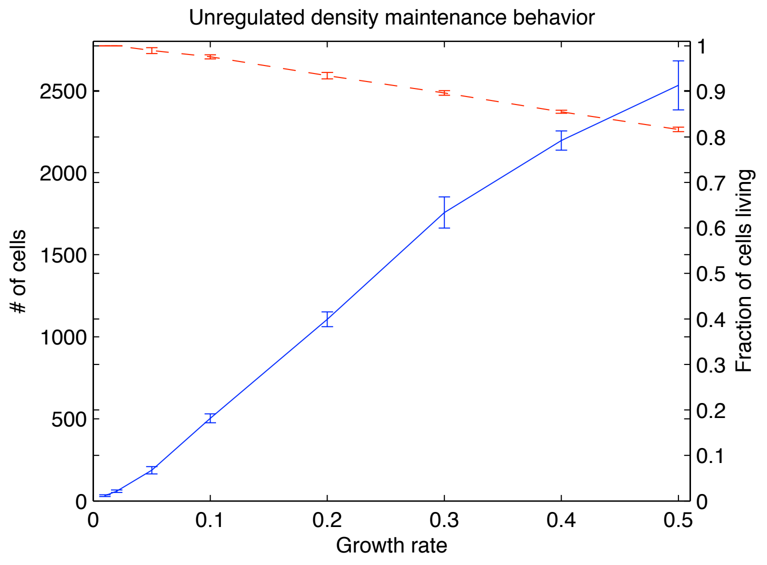


Fig. 3 Behavior of density maintenance and growth system with respect to the growth rate parameter p_d. The number of cells after 200s of growth (blue) is small at low p_d, but the fraction of cells surviving (red) begins to drop at high p_d. Thus the best system behavior is at moderate p_d, but viable behavior spans a large range of values.

beginning with 10 cells distributed in a volume 10 units square and continuing for 200 simulated seconds. As the growth rate p_d rises, the final number of cells in the system (blue solid line) rises sharply, but the fraction of cells that die rises as well (red dashed line shows surviving cells). An intermediate level in the range 0.1 to 0.3 appears to offer the best trade-off, with graceful degradation as the parameter moves away from that level.

4.2 Vascularization

Oxygen delivery by a vascular system requires that there be a capillary vessel relatively close to every cell. When this is not the case, the cell becomes stressed by lack of sufficient oxygen—a graceful degradation situation since the cell does not die. The simple vascularization system here measures stress by distance to the nearest vessel:

```
(def vascularize (source service-range p_v)
    (rep (tup vessel served parent) ; three state variables
         ;; source starts as a vessel; everthing else does not
         (tup source source (if source (mid) -1))
         (mux source
            (tup 1 1 -1) ; source is always a parentless vessel
```

```
               (let ((service (< (gradient vessel) service-range))
                     (server (gradcast vessel (mid)))
                     (children (sum-hood (= (mid) (nbr parent))))
                     (total-children (tree-children parent)))
               ;; adjust radius, for visualization
               (radius-set (mux vessel
                     (* 0.5 (sqrt (+ 1 total-children))) 2))
               ;; grow/shrink vessel network
               (mux vessel
                     ;; if a cell is already a vessel, then:
                     (mux (or (muxand
                          ;; if there is too much branching...
                          (any-hood
                             (and (= (nbr (mid)) parent)
                                  (> (nbr children)
                                     (mux (nbr source) 6 3))))
                             ;; ... and there is no smaller branch
                             (not (any-hood
                                  (< (nbr total-children)
                                     total-children))))
                             ;; or if path to source is broken
                             (not (any-hood
                                  (and (nbr vessel)
                                     (= (nbr (mid)) parent)))))
                      (tup 0 1 -1) ; ... then vessel is discarded
                      (tup 1 1 parent)) ; else vessels stay fixed
                     ;; if a cell is not a vessel, then:
                     (mux (muxand (muxand
                          ;; if some neighbor is a vessel
                          (any-hood (nbr vessel))
                          ;; and there is need of more vessels
                          (dilate (not served) service-range))
                          ;; and a coin-flip comes up heads
                          (< (rnd 0 1) p_v)))
                      (tup 1 1 server) ; then become a vessel
                      (tup 0 service -1))))))) ; else stay non-vess
```

where `tree-children` is a simple program counting each cell's number of descendants in the vascular tree:

```
 (def tree-children (parent) ; given each cell's parents:
   (rep nc          ; 'nc' is the estimated number of children,
      0             ; which starts each cell as a leaf
      (sum-hood                    ; summing over neighbors,
        (if (= (nbr parent) (mid)) ; if a neighbor is a child,
            (+ 1 (nbr nc))         ; add it and its chilren;
            0))))                  ; otherwise, nothing
```

Every cell tracks whether it is part of a vessel and, if not, whether it has *service* from a vessel within the *service-range*, In the beginning, only the *source* cell(s) are part of a vessel. Later, the incremental program adds or removes cells from vessels to incrementally adjust the network. Cells join a vessel at a growth rate p_v when adjacent to a vessel and in range of an unserved cell. Vessel cells dedifferentiate

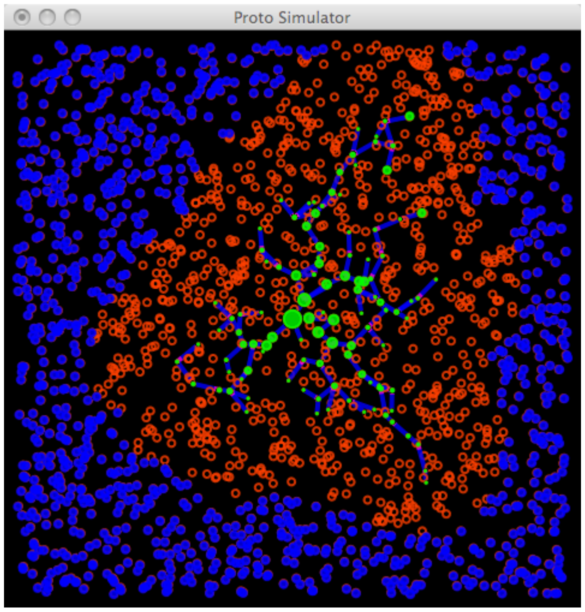


Fig. 4 Vascularization: simulation of a vascular network (green dots, blue lines) expanding the area of oxygenation (red) into a sheet of underoxygenated cells (blue). The area of a dot is proportional to the size of the network descending from it.

when they lose their connection to the source or when too many other vessel cells share the same junction.

Figure 4 shows a snapshot of typical simulated behavior, visualized using:

```
(def drawvasc (v)
  (let ((vessel (1st v)) (served (2nd v)) (parent (3rd v)))
    (green vessel)              ; green for vessel cells
    (blue (not served))         ; blue for oxygen-starved cells
    ;; line to vessel cell's parent
    (sum-hood (* (= (nbr (mid)) (3rd v)) (nbr-vec)))))
```

Figure 5 shows the vascularization system in experiments where the growth parameter p_v ranges from 0.001 to 0.05. For each parameter value, 10 trials were run, where the network is grown for 200 simulated seconds from a seed point in the middle of a network of 2000 devices and devices are distributed on a square 300 by 300 units with a vascularization service range of 50 units. The higher the growth rate p_v, the faster vascularization proceeds and therefore the larger an area is served (blue solid line). The faster vascularization proceeds, however, the more redundancy there is in the system, as reflected by the fraction of cells designated as vessels (red dashed line).

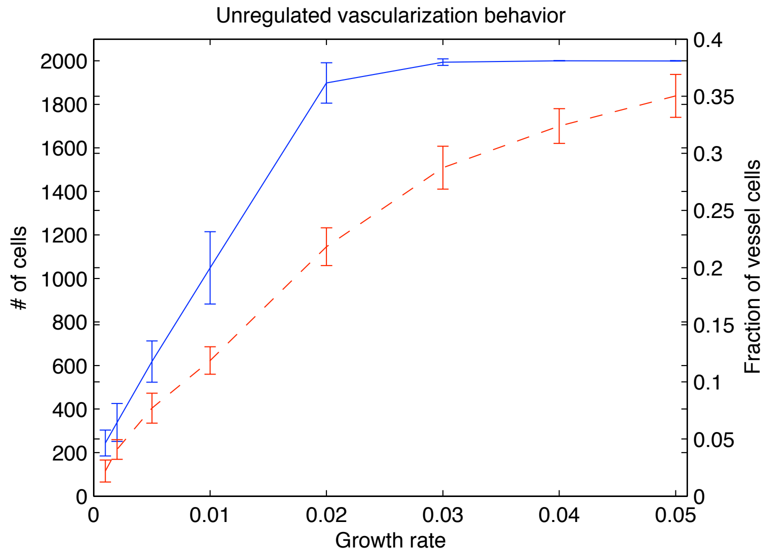


Fig. 5 Behavior of vascularization system with respect to the growth rate parameter p_v. The oxygenated area (blue solid line) expands slowly at low p_v, but the fraction of cells incorprated into vessels (red dashed line) is large at high p_v. Thus the best system behavior is at moderate p_v, while viable behavior spans a large range of values.

4.3 Composite Behavior

These two subsystems can be linked together into a simplified model of tissue growth by the simple expedient of enabling growth in the cell density system only for those cells served by vascularization:

```
(def tissue (src pd pv)
  ;; run vascularization and density maintenance
  (let* ((v (vascularize src 50 pv))
         (dir (cell-density (2nd v) 0 pd)))
    ;; allow all but vascularization source to move
    (if (not src) (mov dir) (tup 0 0 0))
    ;;  visualize vascularization
    (drawvasc v)))
```

Having implemented these two subsystems using functional blueprints, this simple coupling suffices for them to regulate one another into synchronized growth (Figure 6).

Figure 7 compares regulated behavior (blue line) with unregulated subsystems (red dashes), showing a smooth shift in regulatory dominance of the coupled system as p_d and p_v are varied. For each set of parameter values, 10 trials were run, beginning with 10 cells distributed in a volume 10 units square and continuing for 200 simulated seconds. In Figure 7(a), p_d is varied from 0.01 to 0.5 as above, while

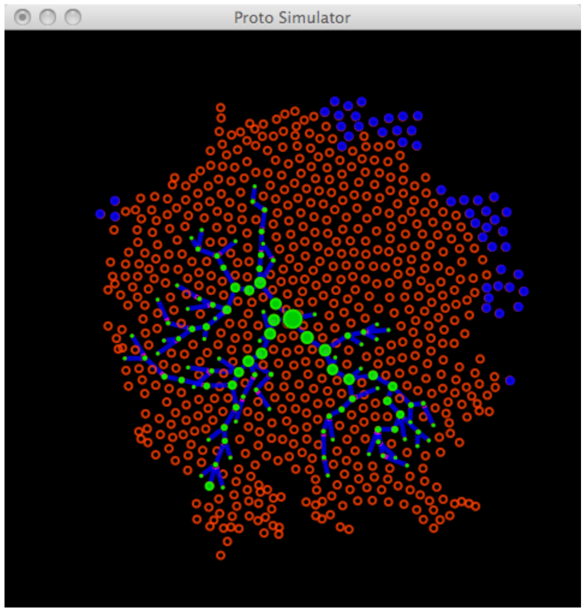
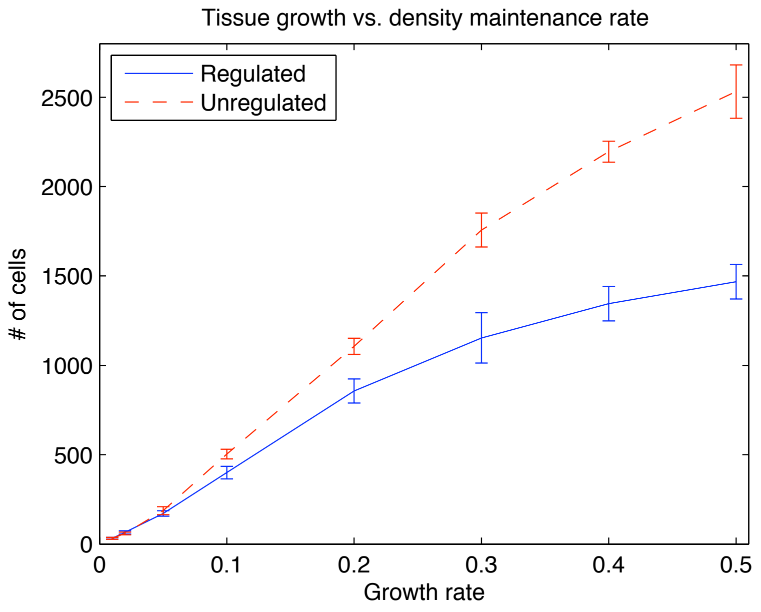


Fig. 6 Synchronized growth of a tissue: growth from cell density maintenance is enabled only for cells served by vascularization.

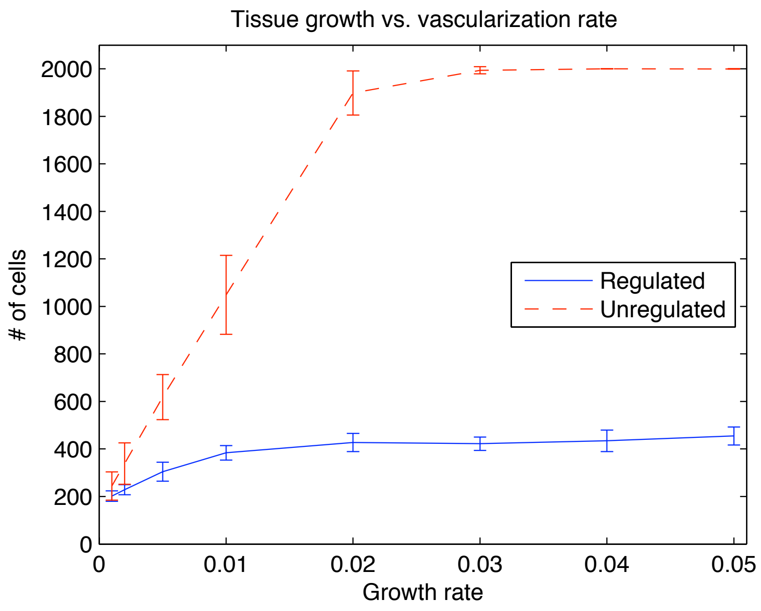
p_v is held constant at 0.02. At low values of p_d, growth from density maintenance dominates, but as p_d rises, cells spread outward faster and their growth begins to be checked by the rate of vascularization instead. In Figure 7(b), p_v is varied from 0.001 to 0.05 as above, while p_d is held constant at 0.1. At low values of p_v, vascularization is the limiting factor, but by $p_v = 0.02$ the limiting factor has shifted from density maintenance to the rate of growth.

This composite system may itself be viewed in terms of a functional blueprint, as these results illustrate, where both density and vascularization are being maintained in the face of stress, and the failure of either checks the progress of the other. Moreover, just as the cell density subsystem was modulated to form a growing tissue, so may the tissue be modulated to grow complex shapes. This can be done by modulating either the cell density subsystem or the vascularization subsystem. For example, Figure 8 shows the result of constructing a letter "F" through regulating cell density (Figure 8a) and through regulating vascularization (Figure 8b).[3] Moreover, the functional blueprint separates the result of modulated tissue growth from the details of its execution, as illustrated by the equivalent constructions in Figure 8a and Figure 9.

[3] For simplicity in this demonstration, the "F" bounds are set by external localization, though they could be self-organized with various methods (see [9], [11]).

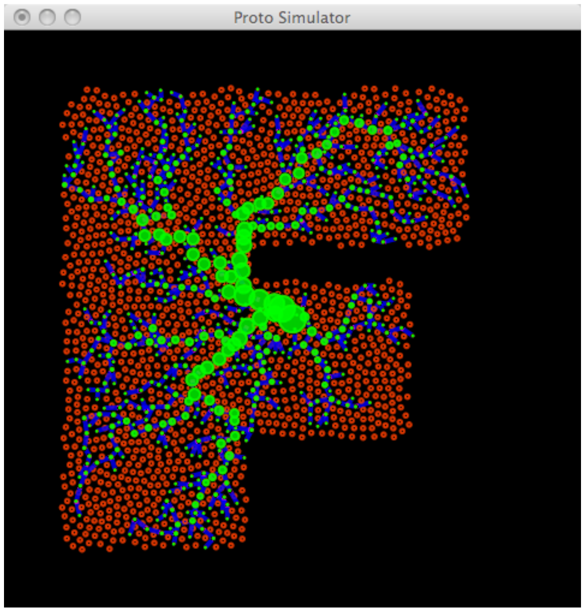


(a)

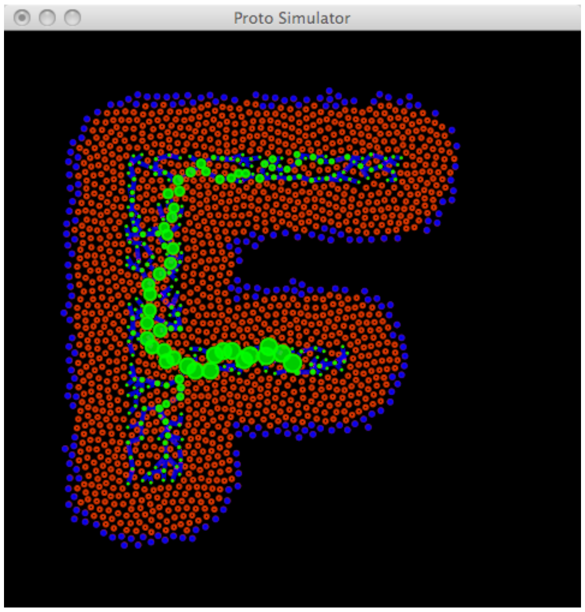


(b)

Fig. 7 Linking density maintenance and vascularization results in synchronized tissue growth, with either subsystem able to regulate the behavior of the composite system.

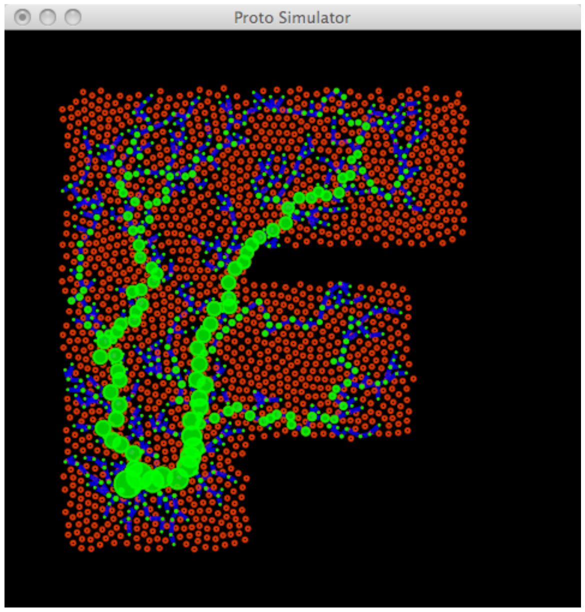


(a)

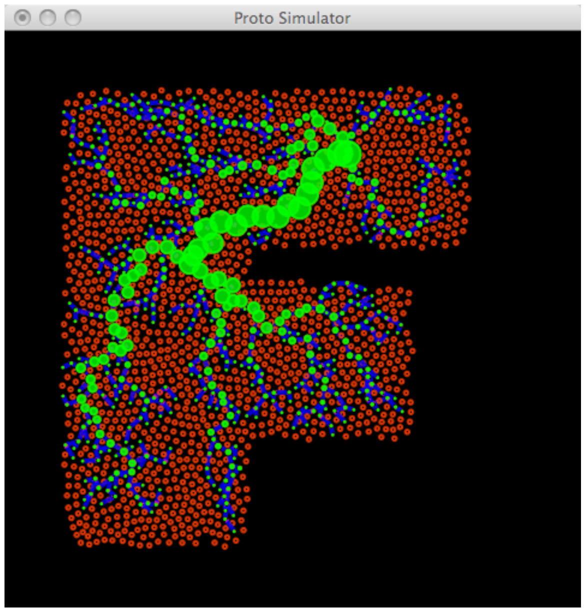


(b)

Fig. 8 The tissue growth system can be modulated to produce complex patterns, such as the letter “F”, by modulating growth of either the cell density (a) or vascularization subsystems (b). In both cases shown here, the letter grows from a seed near its center.



(a)



(b)

Fig. 9 A functional blueprint separates the result of modulated tissue growth from the details of its execution, as shown by equivalent constructions of the letter "F" when grown from a seed in the lower right (a), upper left (b) or center (Figure 8a).

5 Summary

As we have seen demonstrated, a *functional blueprint* approach can be used to create grown systems that are dynamically integrated and able to smoothly transfer regulatory control across regimes. These systems can be interconnected to form composite systems with the same properties of dynamic integration and smooth transfer of regulatory control.

As yet, the functional blueprint approach has not been thoroughly explored, and its capabilities and limitations have not yet been established. However, the simplicity of creating and integrating the models such as those we have considered here indicates good potential for further development. The decoupling of ultimate structure from developmental program offers a possible path to more adaptivity in engineered systems as well as stronger biological models for evolvability and phenotypic adaptation.

References

1. Aubin, J.P.: Viability theory. Birkhauser (1991)
2. Basu, S., Gerchman, Y., Collins, C.H., Arnold, F.H., Weiss, R.: A synthetic multicellular systems for programmed pattern formation. Nature **434**, 1130–1134 (2005)
3. Beal, J.: Functional blueprints: An approach to modularity in grown systems. In: International Conference on Swarm Intelligence (2010)
4. Beal, J., Bachrach, J.: Infrastructure for engineered emergence in sensor/actuator networks. IEEE Intelligent Systems pp. 10–19 (2006)
5. Beal, J., Bachrach, J.: Programming manifolds. In: A. DeHon, J.L. Giavitto, F. Gruau (eds.) Computing Media and Languages for Space-Oriented Computation, no. 06361 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany (2007)
6. Carmeliet, P.: Angiogenesis in health and disease. Nature Medicine **9**(6), 653–660 (2003)
7. Carroll, S.B.: Endless Forms Most Beautiful: The New Science of Evo Devo and the Making of the Animal Kingdom. W. W. Norton & Company (2005)
8. Coore, D.: Botanical computing: A developmental approach to generating inter connect topologies on an amorphous computer. Ph.D. thesis, MIT (1999)
9. Doursat, R.: The growing canvas of biological development: Multiscale pattern generation on an expanding lattice of gene regulatory networks. InterJournal: Complex Systems **1809** (2006)
10. Kirschner, M.W., Norton, J.C.: The Plausibility of Life: Resolving Darwin's Dilemma. Yale University Press (2005)
11. Kondacs, A.: Biologically-inspired self-assembly of 2d shapes, using global-to-local compilation. In: International Joint Conference on Artificial Intelligence (IJCAI) (2003)
12. MIT Proto. software available at `http://stpg.csail.mit.edu/proto.html` (Retrieved March 14, 2010)
13. Nagpal, R.: Programmable self-assembly: Constructing global shape using biologically-inspired local interactions and origami mathematics. Ph.D. thesis, MIT (2001)
14. Prusinkiewicz, P., Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer-Verlag, New York (1990)
15. Shetty, R.P., Endy, D., Thomas F Knight, J.: Engineering biobrick vectors from biobrick parts. Journal of Biological Engineering **2**(5) (2008)
16. Spicher, A., Michel, O.: Declarative modeling of a neurulation-like process. BioSystems **87**, 281–288 (2006)

17. Werfel, J.: Anthills built to order: Automating construction with artificial swarms. Ph.D. thesis, MIT (2006)
18. Werfel, J., Nagpal, R.: Collective construction of environmentally-adaptive structures. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007) (2007)