Cellular-Satellite, a Different Kind of Final Frontier

James Lyke
Air Force Research Laboratory, Space Vehicles Directorate
AFRL/RVSE; (505) 846-5812

Robert Pugh
Think Strategically!
PO Box 14951, Albuquerque NM 87191; (505) 331-1189
Bob.Pugh@Think-Strategically.com

ABSTRACT
Extending the ideas of reconfigurable components and self-organizing, plug-and-play systems offer some very intriguing prospects for rapid satellites of the future. In this talk, we describe the “cellular satellite” as a paradigm for the ultimately modular and rapidly formed system. In this vision, all systems are formed as a vast ensemble of black-box cells, each of which is individually customizable to collectively form an integral system. Locally (within a given cell), all significant functions and properties (e.g., electrical, thermal, and mechanical) are software-definable, similar in principle to the approaches used in ordinary field programmable gate arrays but extended far beyond digital building blocks. Systems are “merely” ensembles of cells whose arrangements are also soft-defined, leading to a DNA-like analogy in which a very long Boolean string, in effect, forms a unique specification of a constructible spacecraft. Initial attempts at cellular satellites would proceed with cells that are physically large (in some sense, the panels of our plug-and-play satellite could be viewed as very large cells), but would evolve through a Moore’s Law-like principle to become eventually tiny smart particles. We believe these ideas provide the ultimate solution for responsive space, in which containers of such “programmable matter” could be rapidly configured with powerful computer-automated design tools to form complex shapes, structures, and systems.

INTRODUCTION
Complex systems are comprised of many components, which are sometimes themselves composed of many components. This recursion, like a hierarchy of Russian dolls, proceeds inexorably to some stopping point, which is a set of primitive elements that are the building blocks of some “universe” that spans all possible compositions of the blocks. We are surrounded by examples: organisms are made of cells, digital systems are comprised of logic gates, etc. While we might view the primitive elements of digital systems and organisms as “atomic,” they are themselves divisible into another recursion of primitive elements. Ultimately (or not), we have the actual atoms that comprise the molecular structures that underlie all of these things. But the notion of atomicity is not limited to material things, but can be ascribed to things without substance, such as language, which is eventually decomposable into letters.

We consider that in each of these examples, the primitive elements are so fundamental that we can ignore their inner structure for the most part, and yield all useful function and structure by simply focusing on the arrangement of elements. An adder circuit is assembled from “AND” and “OR” logic gates, and while these logic gates are comprised of specific arrangements of transistors, we care very little about the role of the transistor when we design the adder. The idea that a relatively small number of simple building blocks are capable of complex expression is something we take for granted. We realize that every object we can touch, hold, bend, melt, or otherwise harness for our purposes is based on compositions of a small number of atoms (~100) drawn from a “library” of elements (i.e., the periodic table). Yet we do not view doughnuts, wrenches, salamanders, and bowling balls as being composable from a common set of primitive elements. When we use a nut and bolt to secure the leg of our new barbecue grill, we do not worry about how the atoms interact to keep the leg from wobbling. Even in a more confined universe, say that of a spacecraft, we do not view telescopes as being built from the same building blocks as communications antennas, solar panels, or flight computers, but they could be.

We believe that it is possible to pursue a fundamentally different way of building systems, a cellularity motif. The cellularity motif is an approach to “extreme modularity,” in the same way that cells constitute a
“modular” approach to creating biological organisms. In the CellSat approach, vast ensembles of tiny, cellular modules would self-organize to form a spacecraft. The individual cells of a CellSat would be drawn from a small library of reconfigurable modules, sort of an extreme version of micro-LEGOs (see Figure 1), functionalized through software only commands. Systems are then configurable ensembles of configurable components, and an overall DNA-like string of information constitutes a constructible specification. Initially, these “cells” would not be of the same dimensional order of cells in the human body, but might through a Moore’s Law-like\(^1\) principle be reduced in size over time to an effectively continuum limit.

Figure 1: Conceptual “LEGO-ized” space platform
(from cover of National Defense Magazine [2])

In this paper, we consider the possibilities of a cellular basis for complex, human-created systems, focusing on spacecraft. This paper is organized as follows. We first describe the concept and consider the incentives for pursuing a cellular motif. Next, we discuss the technological influences that are leading us to consider a cellular motif. We then consider the more detailed template of cells that might be used to form cellular systems such as a CellSat. We finally consider a simple reference architecture and possible roadmap for CellSat research and development.

WHAT WOULD A CELLSAT BE…(?)

CellSats would represent the application of cellularity motifs to a particular type of platform (spacecraft) within a larger family (aerospace systems). While we will restrict the present discussion to spacecraft, there is otherwise nothing fundamentally preferable about cellular implementations of spacecraft vis-à-vis other systems—many other platforms are viable candidates for cellular implementations. (For the purposes of narrative convenience, we may apply the present tense to these ideas, even though CellSats do not yet exist.)

A CellSat is a satellite built from artificial cells that have physical form, function, and a commutable logical framework used for commanding configuration and sometimes to extract data. Cells within a CellSat are “atomic” in that they are not meant to be divisible. They have locally configurable properties/functions that could be as simple as a programmable resistance or surface reflectivity or as sophisticated as a configurable storage, computation, or transmission facility. The dimensional order of cells is expected to be initially “golf ball-sized,” with the expectation of becoming much smaller over time, as new generations of technology emerge. The cells can be identical (homogeneous) or different (heterogeneous) and can be connected in physical/functional arrangements that are suitable for custom missions. Cells would communicate predominately with connected neighbors, sharing or transferring data/commands. More ambitiously, neighbor cells could transfer/modulate electromagnetic signals (including light), heat, or fluid.

In the CellSat vision, systems would be formed based on an automated design and assembly process (aka, pushbutton toolflow) in which a DNA-like representation of a spacecraft is produced in a hierarchical set of configuration files. This “DNA” amounts to an aggregation of local and global rules that form configuration files for dozens to thousands of pre-built cells. A low-level ensemble of configured cells, which we will call a “compound element,” is not simply an agglomeration; that is, the whole is far more than the sum of its unconfigured parts. Each compound element is an evolved hierarchy, just as a word is an evolved hierarchy of letters of the alphabet and is itself a compound element of a sentence and sentences in turn are used to construct a novel. Rules and properties can be ascribed to these compound elements to assign context to the low-level ensembles, at least as a general principle (we can imagine cases where systems function as collectives, like cellular automata, in the presence of only local rules). This configuration procedure continues hierarchically, with the spacecraft platform being the final evolved stage in the hierarchy (also having its global rules and properties). The toolflow specifies the quantity and mixture of CellSat elements and their relative arrangement to form a 3-D system. Each cell in the arrangement is bound to a specific configuration file. The configuration file specifies local

\(^1\) Moore’s Law is an empirical observation made by Gordon Moore that the number of transistors in a given area doubles every 18 months through technological advancement [1].
reconfigurable properties and connective rules. The toolflow also provides the aforementioned hierarchical cues, the rules and properties assigned to each hierarchical level of compound elements (i.e., the sentences, paragraphs, etc. from our novel analogy).

Extreme cases require the configuration files to be unique to each cell. In other words, in a hundred identical cells, it is possible there are a hundred configuration files—each cell requires a complicated and unique description and configuration. However, it is likely that patterns will exist in the way cells combine to form a complex system. Maybe, for example, one hundred identical cells form an inert cube structure. In that case it seems likely that a simpler rule set might govern the formation of corner, edge, surface, and interior regions. The descriptive (“Kolmogorov”) complexity suggests there exist cases where we could reduce the variation in rulesets.

The protocols for self-assembly are established in this manner. Whether in fact the cells are equipped with locomotive capabilities (such as the PARC modular robots [3]) or not may be irrelevant, at least in ground assembly where humans or robots can provide the locomotion necessary for guided self-assembly [4]. Systems based on the CellSat concept might choose to adopt the cellular strategy in whole or in part, in the latter case building onto an existing (non-cellular) system as a substrate. Pre-built (non-cellular) elements like payload telescopes and cameras might exist as “inclusions” of a cellular system, around which the rest of the CellSat is “poured.”

…and why would we want it?

Given the overhead and extra initial development effort needed to create a cellular infrastructure, why would we want to make such a contrivance? We believe there are several strong benefits to be derived from creating the CellSat infrastructure—most relate to the benefits inherent in reconfigurable systems.

Speed. The first benefit is the reduction of time. In archetypical reconfigurable systems, such as field programmable gate arrays (FPGAs), reconfigurability amounts to software-definability. FPGAs are in effect digital systems in which the configuration of logic, interconnect, and memory are controlled as millions of individual “0-1” programming decisions that ultimately shape arrays of pre-built silicon resources into a system equivalent to one created in custom silicon from scratch. Since the configurations are soft-defined, implementing systems using reconfigurable elements can be done much more swiftly than building custom versions of the same elements. Components integrating deeply-intercalated reconfigurability features can be pre-built, and inventoried, then, when needed, pulled off the shelf and configured for immediate use. How far can these benefits be driven? At one level, we might argue that the CellSat can become the ultimate response to the problem of Operationally Responsive Space (ORS), in which it is desirable to create and launch spacecraft in the shortest possible timeframes. It is conceivable that in the future bins of “blank” cells might be held in storage for creating nearly “instant” systems. When needed, an automated design and assembly console (described later) will create a DNA-like configuration file to guide the self-assembly of hierarchical cellular ensembles that produce a field-ready system. Certainly, this seems more the domain of science fiction (movies like Terminator 2 come to mind), but many of the requisite elements of such “instant system” technologies do not seem so distant.

Robustness/resilience. Design errors are common in complex systems no matter how carefully they are created. In custom, monolithic systems, errors can be difficult to rectify. In modular, reconfigurable systems (such as the CellSat) errors in design can in many cases be eliminated through reconfiguration with much less impact, even if the system has been deployed in the field. Beyond the offline discovery and rectification of errors, CellSats can permit the possibility of autonomous error correction. CellSats would enable new, self-healing protocols. It is conceivable that unreliable cells can be circumlocuted or expunged from a complex system in the field, with new cells (possibly in reserve) inheriting the “rule sets” of displaced cells.

Flexibility. Finally, the flexibility of CellSat itself may be its greatest benefit. The potential flexibility comes in different forms through the life cycle of a CellSat. During initial creation, it is possible to reconfigure individual cells to support enhanced diagnostics for test/debug. Temporary probe wires for example might be formed through opportunistic chains of cells to access buried portions of a CellSat and then be “dissolved” when no longer needed. Satellite or subsystem designs can be modified if requirements change during development, as is often the case. Of course, flexibility in fielded / deployed systems is always a desirable property. The ability to remotely “alter the mission footprint” of an orbiting system (“tele-alteration”), even autonomously, can mitigate the need to field another space system. Certainly, it is conceivable that the needs giving rise to the deployment of a system can change, further emphasizing the desirability of a system to have the flexibility to adapt to those change needs. CellSats offer an interesting possibility in the final stages of a system’s life cycle in which, rather than deorbiting a spacecraft, it is possible to consider the reclamation of its resources. In these
cases, the cellular resources of one system can be disassociated and made available for use by other orbiting platforms. CellSats need not be destroyed through deorbiting, at least not in total, but instead might be recycled indefinitely.

While these benefits are compelling, we must demonstrate not only that this architectural approach is feasible at some basic level, but that the overhead associated with it is tenable for a useful class of missions. This paper attempts to address the first issue in hopes of motivating practical implementations. These implementations, combined with technology advancements, will be necessary to address the latter concern.

BACKGROUND

In this section we chronicle a number of the underlying research influences that have percolated over time to form the basis of the CellSat concept

Related AFRL Research

Many of our most-prized innovations have emerged as responses to frustrations encountered in our research and experimentation efforts at the Air Force Research Laboratory (and its precursor organizations). We cannot always account for why things go wrong when we set out to do world-class research, but we have come to embrace the idea that failures can be “information rich” and appreciate such creative outlets as oft-underestimated silver lining. For example, much of the body of our plug-and-play research [5] emerged as we tried to cope with bad side effects in space experiments, deficiencies that we saw in other research ideas (e.g., research in multifunctional structures [6]) and limitations to ideas that were just not quite good enough in the day they were first explored (such as multichip modules). As the world continues to wait for molecular electronics to arrive (apparently some of us thought they were imminent nearly 50 years ago [7]), we have seen the pursuit as an incredible idea engine where almost every concept, no matter how bad, might just have a small filament of plausibility and make a major contribution. More than any discipline that either of the authors have been involved with, it is the pursuit of the very smallest things that has emboldened us to “dream big” (we think Feynman would have liked CellSat).

Advanced packaging. We studied the art and science of advanced microelectronics packaging in pursuit of density and performance improvements beginning in the mid-1980s, originally in support of radiation-hardened space systems for the Strategic Defense Initiative Organization. We developed and demonstrated through our industry partners a great variety of two- and three-dimensional multichip module (MCM) technologies which in some cases represented very tantalizing benefits, such as hundred-fold density improvements in the volumetric packing efficiency of integrated circuits compared to conventional approaches of that era. Unfortunately, our research tended to be somewhat fragmented due to funding constraints, meaning that we might miniaturize a computer module for one customer and a memory design for another customer. When we would have the occasion to combine the research products of different activities together, we found the results disappointing, as depicted in Figure 2. In this depiction, “A” and “B” represent electronic functions to be miniaturized. In a conventional development (Figure 2a), the combination of A and B would occupy an amount of space slightly greater than the linear combination of the space of the individual modules. Through miniaturization efforts, the functions A and B could be compacted by significant fractions, as shown in Figure 2b. Combining the function A from one research program with function B from a second research program would usually produce the disappointing result depicted in Figure 2c in which many of the gains obtained by piecewise miniaturization are lost when those pieces are combined. The source of the problem (the cloud in Figure 2c) is a problem we might call the “tyranny of interfaces.” Disparate interfaces in random components require the addition of glue hardware and software. Such became the frustration of our early packaging research, impressive but not practical in real systems for the most part.

![Figure 2: The Tyranny of Interfaces](image-url)

**Multifunctional Structures.** Even as we struggled through a number of challenging MCM research projects, in the mid 1990’s a parallel research program called “multifunctional structures” (MFS) was initiated to study the merging of function and structure in
spacecraft. Once again in the name of improved miniaturization, MFS programs sought to eliminate complex wiring harnesses by pre-integrating flex circuitry into structural panels [8].

In theory, the idea seemed appealing. Panels and chassis elements needed to form the supporting structure of a spacecraft would serve as the opportunistic conduits for wiring, thermal management, and other spacecraft functions. Thus, MFS was also an advanced packaging program, operating at a higher level of the spacecraft-system. It was compared to the LEGO building bricks used in children’s play. Implicit was the suggestion that with MFS, it would be “child’s play” to build a spacecraft. In a cruel twist, reality intervened, once again, in the form of the “tyranny of interfaces.” Unfortunately, the simple addition of flex circuitry did not resolve the problems of disparate interfaces between satellite components. Once again, the lack of any unified architecture strategy defeated the hopes in MFS approaches to effect any meaningful reduction in size and complexity of space systems.

While it was possible to laminate printed wiring in the form of flexible circuitry on the surface of a panel, it was not generally possible to avoid messy rats’ nests of cables to accommodate dissimilarities in component interfaces and design changes to the spacecraft system. MFS, like MCMs, were rendered at best marginally effective, usually at a higher cost due to the exotic technologies involved.

Reconfigurability and the pursuit of gluelessness. FPGAs comprise the most successful class of reconfigurable systems and provide many useful insights for extending reconfigurability to non-traditional electrical and other phenomenological domains. FPGAs are capable of implementing very complex digital electronic designs through software configuration of logic, memory, and interconnect resources. Reconfigurability, as epitomized by FPGAs, affords some intriguing possibilities to deal with the “tyranny of interfaces.” We began in research to consider the use of FPGAs as morphable interfaces. By the late 1990’s a growing number of researchers worldwide were touting the benefits of FPGAs as reconfigurable computers in which algorithms could in some cases be sped up 400X compared to a traditional general-purpose computer. While this benefit was indeed attractive, we were more intrigued by the idea of a “malleable signal processor” (MSP), using FPGAs to adapt the interfaces between dissimilar components (such as A’ and B’ in Figure 2c) in hopes of reducing the external hardware (the cloud). We found that we could reduce, but not eliminate, the cloud. This is because FPGAs are generally limited to digital circuitry, and in some cases a lot of the cloud was caused by non-digital circuitry. We began to develop a wish-list that eventually became a roadmap for a universal reconfigurable system.

Level shifters. Sometimes each side of an interface is based on different voltage standards, for example one might need to connect a 5V circuit to a 3.3V circuit. If we put level shifting transceivers in the malleable processor, we could eliminate the need for external components to perform this function. Since the MSP is a multi-purpose device and since it would not be possible to know what type of level shifter might be needed in every application, we would need to use configurable level shifters.

Analog conversion and preprocessing. Of course since the real world is analog, for the most part, it would be necessary to add analog-to-digital convertors to process sensor signals in an FPGA. Analog conversion tends to be very tailored to application, and therefore it would be necessary in general to embed a variety of convertors to support different interface scenarios. Even in the case that one universal conversion circuit could be found, it would be desirable to provide the ability to configure gain, offset, and input filter characteristics. Analog reconfigurable approaches have been pursued electrically [9-10], but they have not gained the phenomenal traction of digital field programmable systems. At the most basic level, the pursuit of digital reconfigurability is simpler than establishing reconfigurable versions of analog components. Analog systems can be roughly divided into three sub-domains: low-level signal, microwave, and power.

Power supplies and power drive. In the event we ever develop a reasonable reconfigurable approach to eliminating the cloud of uncertainty in the interfaces to sensor blocks, we would soon need to consider the actuator interfaces and the power generation and management facilities within systems. Embedding configurable digital-to-analog convertors would be necessary, with configurable power delivery to accommodate a variety of electrical loads. Many application scenarios can be envisioned even for embedded systems, in which wide ranges of power drive (milliwatts to kilowatts) and voltage range might be involved. We may also need to consider varying levels of post-conversion signal processing.

Microwave. To support communication-function interfaces, it will be necessary to implement configurable microwave building blocks in the malleable processor, to support functions ranging from antennas to configurable filters, oscillators, and other
radio-frequency building blocks. Creating reconfigurable versions of microwave components is even more challenging, since in addition to the other limitations inherent in analog components, microwave components are affected by the transmission line properties of the components and their embedded environment. Most work in reconfigurable microwave components has been undertaken at the primitive component level, using technologies such as microelectromechanical systems (MEMS) to create adjustable capacitors, and using switches to construct a variety of microwave building blocks, the stuff of true software-definable radios.

We began to wonder how difficult it might be to draw the boundary of an FPGA architecture differently to enable construction of reconfigurable analog devices. A very simple FPGA, shown in Figure 3, consists of an array of configurable logic blocks (CLBs) and programmable wiring resources. Since a complex digital system can be represented as a network of primitive logic functions, it is straightforward to map a digital system into the 3a “fabric” by choosing a CLB for each primitive function, configuring it accordingly, and then forming the wiring network by closing a suitable sequence of switches (Figure 3b). While the CLBs have a definite interior structure (Figure 3c) we can also view them as black boxes. These black boxes could contain other types of building blocks, including ones with non-digital contents 2. If we were able to create a “super-FPGA” (see Figure 4) in which we extend the idea of the black box to include any electrical component, it would be possible to interface components gluelessly. In other words, the “cloud” in Figure 2c would disappear, since any disparities in interfaces could be compensated by choosing a suitable arrangement of black boxes from the extended palette of options in this “super-FPGA.” In fact, the super-FPGA would be in effect, a parts cabinet “on a chip” capable of directly implementing any electronic circuit, given enough black boxes of sufficient diversity.

Of course, the “super-FPGA” as posed in Figure 4, is not generally practical with today’s technology for many reasons. Semiconductor processes, for example, do not presently provide a rich enough variety of electrical part types to implement arbitrary interfaces. They cannot support the wide voltage and power ranges needed in many real-world electrical circuits. Furthermore, the non-ideality of wiring within an integrated circuit and undesired electrical parasitics would severely limit performance.

---

2 Within complex FPGAs, the sophistication, number, and type of configurable black boxes has grown dramatically, even though they remain predominately digital in nature.

Figure 3: Structure of an FPGA

**Adaptive wiring.** Making a discretized version of the “super-FPGA” might be approachable, however, even if only in a sharply limited application domain and level of sophistication. In this case, one would integrate a number of electrical blocks as discrete components onto a special sort of wiring board, a substrate in which the arrangements of wires would be programmable. We called this idea the “adaptive wiring manifold” (AWM), literally a reconfigurable wiring harness, inspired in part by the “super-FPGA,” in part by MFS, and in part by the advent of microelectromechanical systems (MEMS) based switches, especially versions capable of latching. The AWM concept [11-12] involves the extension of the ideas used in FPGA routing networks, except rather than monolithically integrated transistors, the AWM would employ latching MEMS switches to piece together segments of wire to form a desired pattern (called a netlist). By using physically larger, latching switches, it would be possible to form persistent copper pathways in wiring systems capable of accommodating the distribution of
power, analog, and other discrete signals between spacecraft components. Such a strategy would realize the promise of MFS by creating a smart substrate that could be fabricated, stockpiled, and retrieved for just-in-time configuration as a panel with built-in wiring. The ideas of AWM were sufficiently well-evolved to the point that two generations of demonstrations were constructed [13].

**Plug-and-play research.** With reconfigurability we explored techniques and tools for coping with the “tyranny of interfaces.” With reconfigurability, we introduced software-definable features and functionality that could at one level be used to carry out mission-related functions and at another level could be used to resolve discontinuities at the interface. Reconfigurability introduced a great many degrees of freedom. Plug-and-play provided the disciplines for managing these degrees of freedom, and, in the process we feel, providing an effective mitigation of the “tyranny of interfaces.”

“Plug-and-play” is a much abused term, especially in aerospace, as a property hoped for but rarely achieved. We studied the phenomenon intensely, in support of an AFRL study of ORS that originally focused on the problem of responsive launch, but was later broadened to encompass spacecraft. We began to coin terms such as “six-day spacecraft” to frame the challenge in bold strokes and terms such as “responsivionics” to denote a type of electronics that allowed systems to be built quickly. We slightly obsessed over why complex systems were complex in the first place and what could be done about it. We drew inspiration from FPGAs (which could be swiftly configured compared to a built-from-scratch integrated circuit) and electronic design automation (EDA – the process used to design vastly complex, i.e., millions of transistors, integrated circuits and FPGAs). The EDA industry dealt with fiercely competitive product cycles in which delays of days or weeks could be catastrophic to product rollouts. We also drew inspiration from the commercial PnP technologies in personal computing. The resulting synthesis is referred to as “space PnP avionics” (SPA), consisting of the following key technologies [14]:

**Self-describing components.** SPA components (devices prepared with a SPA interface) are abstracted in electronic descriptions, referred to as extended Transducer Electronic Data Sheets (xTEDS) that are embedded within the components. They amount to a machine-readable interface control document (ICD).

**Standard interfaces.** A number of well-known data-transfer interface technologies were studied to form the basis of SPA. USB became the basis of SPA-U, spacewire became the basis of SPA-S. These interfaces combined the data infrastructure with power delivery and clock synchronization.

**Appliqué sensor interface modules (ASIMs).** As the design of commodity USB peripheral designs benefit from the use of pre-designed USB interface chips, we created the concept of interface modules to simplify the task of creating SPA components. ASIMs make simpler both the task of creating new SPA components from scratch, as well as legacy conversions (adapting non-SPA components to become SPA components).

**PnP middleware.** Inherent in any plug-and-play approach is the ability for networks to auto-form (self-organize) as devices are added and for components to be able to find other components. The PnP middleware, referred to as the Satellite Data Model (SDM) provides the software infrastructure to enable pre-built hardware and software components (both have xTEDS) to advertise and exploit services automatically.

**Push-button toolflow.** Automated toolflows enhance the ability of non-expert users to create complex systems through a guided wizard-driven design process that results in the specification of a buildable spacecraft based on PnP components.

**Test bypass.** Most ASIMs feature an auxiliary connection port that enables the direct injection of synthetic sensor data or the interception of actuator data. This test bypass concept can be integrated with simulations to provide a powerful debug infrastructure within the context of a hardware-in-the-loop environment at any stage of integration.

These concepts have been an important set of guiding principles for the creation of rapidly-integrable systems. At the time of this writing, an entire spacecraft is being built based on the SPA concepts. Before PnPSat, the SPA concepts were initially demonstrated in a ground testbed facility (the Responsive Space Testbed at AFRL’s Phillip Research Site, Kirtland AFB, NM) and later in a sounding rocket test at White Sands Missile Range in NM. They have also been integrated in the AFRL TacSat 3 mission in a limited experiment containing four SPA-U (USB-based) ports. PnPSat uses principles of modularity, encapsulation, and complexity hiding to support the rapid construction of a spacecraft from a collection of self-describing SPA components. The panels of PnPSat are hinged, separable, and are themselves SPA components. The “open” configuration of PnPSat is shown below (Figure 5). PnPSat allows for an almost arbitrary arrangement of elements to form a spacecraft. The components of the first PnPSat are physically large and support limited reconfiguration capabilities. For example, there are
currently no provisions for reconfigurable mechanical attachment, electromagnetic waveguide formation, thermal transport, and other functions that will be required in more flexible approaches, as we envision in future generation concepts, such as CellSat.

**Figure 5: The PnPSat in its “Open” Configuration**

**Molecular Electronics.** At the encouragement of computational chemists at our laboratory, we began to discuss strategies to harness reconfigurable computing to address their problems. These discussions eventually evolved from studying computations of molecules to computations with molecules, and we teamed with other research groups and received support from DARPA to develop architectures. It seemed logical that any building block approach for a molecular computer would be necessarily reconfigurable at a finely granular scale. Reconfigurability would become not only the approach of choice for shaping computation within a molecular “fabric” but also a good strategy for dealing with the many defects of the chemical synthetic processes that would create the building blocks, and the self-assembly processes used to assemble the blocks into a computer using vast numbers of these blocks. Single molecules, even of the sophistication proposed or demonstrated by some of the DARPA Moletronics teams [15-16] could not “compute” but required some minimal arrangement of multiple molecules into a primitive compute element that we referred to as a “molecular nanoblock” (MNB). MNBs would rely on chemical self-assembly to aggregate into complex structures (as suggested in Figure 6). Cellular automata concepts provide a rich basis for exploring the templating of MNBs into extended (three-dimensional) molecular integrated circuits. In a second collaborative activity, we teamed with the Naval Research Laboratory, Scripps Institute, and later the University of New Mexico School of Medicine, to study the use of plant and bacterial viruses as the basis of nanoblocks. In this case, the team explored mutations of viruses to essentially encode with RNA the attachment points of the viral capsid (shell) for molecular circuits and wires. The individual virion itself undergoes a remarkable self-assembly (discussed later), and the viruses in nature tend to self-organize as crystalline compositions, providing an excellent periodic template for cellular-based electronic structures.

**Figure 6: Results of Chemical Self-Assembly**

**PROTOSATS AND MACROSATS**

We now describe the conceptual progenitors to the CellSat, referred to as “ProtoSat” and “MacroSat,” which emerged in 2000-2001 [17] from discussions about spacecraft protection as a means of showcasing reconfigurability in future space systems. They draw influences from all of the previously described work. Protosats, referring to primitive spacecraft building block units, would contain a microcosm of a full-sized spacecraft, in the form of a hexagonal tile (see Figure 7). The tile, influenced by MFS, would contain a variety of electronics for processing and storage, solar cells, guidance and propulsion, simple communications, and physical structure having elementary edge connectors with mechanisms to cause attachment when in proximity. The ProtoSat also featured an attachment point roughly centered on the planar surface of the tile to accommodate the attachment of some external payload or other structure. In a sense, the ProtoSat is a satellite in its own right, theoretically capable of implementing a very simple space mission. It was designed, however, to function as part of an ensemble of similar “smart tiles” that would form a deliberate arrangement referred to as a MacroSat. Macrosats would, by their flexibility and extensibility, serve as a basic skeleton or shell of a variety of full-function spacecraft. We emphasize skeleton in recognition that the ProtoSats did not have sufficient functional sophistication to implement large cameras and other payloads with highly optimized structures, ranging from telescopes to propulsion tanks. Macrosats would use the attachment points of ProtoSats to form a scalable pegboard-like grid to accommodate the addition of these elements as separately fabricated pieces that could be attached to the MacroSat after formation of its skeletal structure, presumably to be automatically integrated when plugged into the panels using mechanisms similar to those we later developed for PnPSat.
Macrosats provided some intriguing if not controversial possibilities in the rapid and flexible formation of robust satellites. In principle, ProtoSats could be built en masse and stored on the ground or in orbit until needed. When needed, they could be retrieved, configured, and assembled to form MacroSats on demand. The assembly would normally be performed on the ground, although the idea of orbital self-assembly was considered as a remote but attractive possibility in on-orbit servicing research. Some of our early animations depicted the ability for autonomous self-assembly (similar to the MNBs in solution shown in Figure 8). This suggestion of autonomous self-assembly in space contributed an additional hint of science fiction. Among the issues in such autonomous in situ self-assembly was the need for many piecewise precision navigation operations (even a single pairwise precision docking operation is non-trivial) and the profligate expenditure of precious internal fuel reserves to achieve the necessary maneuvers during self assembly. On these bases, other attractive (and also controversial) features of MacroSats included:

- Self-healing, in which spare ProtoSat tiles could be moved into the positions of damaged ProtoSats as substitutes;
- Self-protection through temporary dissolution of MacroSats into ProtoSats and dispersal to “dodge” threats, followed by a subsequent re-assembly of the MacroSat and reconstitution of its orbital capabilities;
- Reclamation, in which old, un-needed satellites are cannibalized as sources of ProtoSats to assemble new platforms or repair existing ones.

The concepts of self-assembly that pertained to MacroSats became a very interesting research problem. At one level, it is difficult to escape the obvious comparisons to biology, such as the aforementioned viral self-assembly. Viruses, such as the simple icosahedral forms (examples we studied in the molecular research include cpnv and ms2), are created through a seemingly magical process in which a strand of RNA triggers the spontaneous self-assembly of many identical protein sub-units, forming the icosahedral shell of a completed virus (see Figure 9), which encapsulates the RNA as an infectious payload. It is well-known that virus self-assembly is governed through a complex rule set [18]. Is it possible that in an artificial imitation of biology, using completely different building blocks, rule sets might also be applied to man-made building blocks?

**Figure 7: Depiction of a ProtoSat Configuration**

AFRL conducted a rudimentary study to explore artificial guided self-assembly (i.e., using human or robotic assistance to provide the necessary kinetics). For this work, our researchers designed and built several dozen simple, identical square tiles from printed wiring boards. Each of the boards was equipped with a simple microprocessor to process rules as strings of configuration information. The tiles were designed to attach together (using connectors) laterally in each of four Manhattan (90°) orientations and at three edge mating angles (-90°, 0°, 90°). The boards served as simple, inert “ProtoSats.” To form “MacroSats,” an ad hoc design procedure was followed in software to form a target “MacroSat” structure, resulting in the generation of a compound bitstream that specified the string-form information regarding the connective relationships of...
all ProtoSats. This binary specification would be “injected” into any ProtoSat, as they were all initially blank, through a cable connected to a host PC. Starting with an initial program on a seed ProtoSat, a MacroSat would be constructed through a number of game-like moves in which a human or robot would take blank tiles and begin to speculatively place them. In this case, each placement was a “move” in which an edge of a new tile joined the existing assembly. A series of LEDs on the edges of each tile provide visual cues to a human operator based on correct (green) or incorrect (red) results of trial-and-error placements to assist in the self-assembly of a MacroSat. Eventually, when satisfying assignments were made through a number of placement trials on all edges of all tiles, a complex MacroSat shell structure would be realized.

This guided self-assembly approach offered a crude mimicry of biology in that the software string, like DNA, provides a unique description of the entire structure (organism). Unlike biology, the ProtoSats (analogous to cells) are initially blank and can inherit the “DNA” of an infinite number of possible systems (or up to \(2^N\), where \(N\) is the length of the binary specification). We found in early work that separating a “MacroSat” structure into two pieces led to the ability to regenerate two copies of the MacroSat based on playing the aforementioned game with each of the split structures using additional blank ProtoSats. This remarkable feature was unintended, but we disabled it since we did not find it working reliably in the early prototypes. Such protocols, if perfected, give rise to interesting possibilities, not all of them beneficial. The obvious positive benefits include self-healing and self-replication. In the event of structural defects due to damage, it is in principle possible to exploit the innate self-assembly protocols to once again establish a healthy MacroSat configuration through substitution of tiles (tagged as “bad” through an override mechanism) with good tiles. Self-replication might be done by forcing the release of a small portion of the structure manually (a “bud” being as small as a single tile) from a finished MacroSat, which would result in creating a new copy from the bud, while restoring the parent through self-healing. Other implications of these protocols are not beneficial and must be considered. The first of these are mutations in which some form of glitch (such as a single event upset caused by radiation in the natural space environment) morphs the DNA string producing in the worst case an aberrant mutation in a MacroSat. This amounts to a “collision of bitstreams,” a clash between the original programming and an altered version. It is not certain that we can know in some cases, which bitstream is correct, removed from the original reference. As we know, some level of random variation underlies evolutionary processes, and perhaps a controlled implementation of a Darwinian-like variation might have utility. In the extreme case, however, one could envision malicious codes, like artificial cancers, that disrupt MacroSats in subtle or diabolical ways, possibly to eventually repurpose the ProtoSats, cause MacroSats to disassociate, or worse, fashion them into weapons against their creators.

**A REFERENCE CELLSAT ARCHITECTURE**

In this section we describe the CellSat as a second-generation ProtoSat/MacroSat architecture, inspired by much of the previously described work. Probably the central tenet of CellSat is the idea of hierarchical reconfigurability, i.e., a system is a configurable ensemble of configurable components (which themselves may be configurable ensembles of configurable components). Soft-definability is becoming more important in society. Consider recent trends in manufacturing of “smart” products. In the past, if information was added to a product (e.g., an adding machine) it was added during the manufacturing process and the value of the product resulted solely from how it was designed manufactured [19]. Today, for most products, the information is added after the product (i.e., hardware) is manufactured. The information is software. Whether it is the software that makes a home indoor-outdoor thermometer talk wirelessly to your computer, or the “software” (i.e., configuration bitstream) that encodes the functionality of your FPGA-based modem, the information is added after the hardware is manufactured. This approach reduces the manufacturing cost and increases the flexibility of the product. The CellSat concept takes the fullest advantage of the trend to build the hardware (cells) and add the information (DNA) later.

CellSat is a cellular system, which we will define as a purposeful arrangement of protocol-compatible cells. In this section, we will concentrate on cells having a particular size (golf-ball ~ 43mm or 1.7in), not because the size is important, but simply to facilitate the discussion of a simple reference architecture. It is clear that many things can be altered in the architecture, as indeed it would be desirable to reduce the size of the cells, eventually perhaps to an infinitesimal limit. We say “purposeful” to permit the most liberal definition of what collection of objects would constitute something useful.

We start by defining a **cellular system** as a uniformly gridded cellular array being a 1-D, 2-D, or 3-D packed arrangement of identically-sized cells, in which each cell has a template of nearest neighbors. In the case of a 3-D arrangement there are six neighbors altogether: above, below, front, behind, left, and right. In cellular
automata (CA) theory, the template is referred to as a radius-1 \( r=1 \) von Neumann neighborhood, which includes the cell and its neighbors (adding the eight diagonal corners creates a Moore neighborhood). 1-D and 2-D cellular systems obviously have smaller neighborhoods, in general of size \( 2d+1 \), where \( d \) is dimension. Unlike CA, our cells are not mathematical abstractions, but physical units. Each cell contains one or more soft-definable functions and/or properties and a common configuration infrastructure to control these properties. Each cell is uniquely identifiable and can be aware of its immediate neighborhood.

**Cell Structure.** A generic 2-D cell is depicted in Figure 10. It is shown as a symbol (Figure 10a) having four ports, each connecting to a nearest neighbor in a Manhattan direction. While these cells are functionally considered atomic, they have an important interior structure, as shown in Figure 10b. Here, we distinguish four “egalitarian” connection ports, a configuration management infrastructure (nominally identical in all cells), and a payload section. Port details are suggested in Figure 10c. In general, the connection port will have subports: a phenomenology-specific subport (only electrical connections are shown, as a series of wires), configuration subport (shown as four wires), and an attachment mechanism, shown linked with the configuration subport. The configuration subport and the controlled mechanical attachment constitute undesirable overhead, necessary for configuring the payload and phenomenological sub-port connections to neighbors.

**Example of simple cellular system.** It is perhaps at this point best to use a simple example to make plain these working definitions. We will describe a simple configurable wiring manifold based on a 2-D cellular array. In this case, the cells implement wiring patterns based on connecting wires within cells. An example of such a wiring problem being solved using cells based on this type of cell system is shown in Figure 11. In the figure, a 4x4 array of wiring cells is depicted, with north, east, west, and south edges being defined as simple vectors \( y, x, v, \) and \( u \), respectively. The wiring problem is specified as the netlist\( \{ \{ y_1, y_2 \}, \{ x_1, v_2, v_3 \}, \{ y_1, u_2 \} \} \). One (non-unique) solution is shown based on a configuration of configurable cells, a very simple cellular system.

![Figure 11: A Simple Configurable Wiring Manifold](image)

More details about the cells used to make up the Figure 11 system are shown in Figure 12. The symbol for the cell (Figure 12a) is a building block for a scalable wiring network, such as might be useable to form a configurable wiring harness. In this case, we will define our payload (Figure 12b) to be a very simple switch matrix, whose only function is to allow any single electrical signal from each of its ports to connect to another port. The configuration port receives

![Figure 12: Building Block for Scalable Wiring Network](image)

\( ^3 \) A netlist is simply a list of termini sets that are to be interconnected.
commands percolated through the mesh to configure the individual switches within a cell as needed. A partial schematic is suggested in Figure 12c, demonstrating the linearization of the switch matrix through a shift register in which individual bits from a Boolean string provide the configuration of the matrix.

Based on these notions, we can in a very simple manner define the payload bitstreams of Figure 12 based on the Figure 12c shift register mapping. For example, the row 1, column 1 cell bitstream must connect its west terminal to its south terminal and simultaneously connect its north terminal to its east terminal, which is implemented through the partial bitstream $σ_1σ_2σ_3σ_4σ_5 = 100010$. The reader may easily find a number of alternate bitstreams that implement the equivalent functionality. It is possible to show some robustness features by considering how to compensate for a defective cell, such as the case depicted in Figure 13. This figure illustrates self-healing by circumlocution, which is based on reconfiguration, as opposed to healing by cell replacement where the defective cell would be replaced with a good cell and the same bitstream could be used.

![Figure 13: Example of Self-Healing by Circumlocution](image)

**Cell templates and types.** It is clear through the preceding example that a number of implementation details must be set in the architecture to even begin a discussion of cellular systems. In that example, we chose a minimal number of port conductors (i.e., one pin instead of five per port), the dimensionality ($d=2$), and limited the phenomenology to electrical wiring. In fact, the cells in this example had “non-expressive” payloads, meaning that the utility of cells was defined as only facilitating the relationships between cell ports. As a simple example of an “expressive” payload, consider the concept of a pixellated antenna, shown in Figure 14. In this example, the effective surface of a cell can be set in either of two states, insulative (zero conductivity $σ$, free-space impedance $v_0$) or conductive (infinite conductivity). We also define for this example that neighbors in a $r=1$ Moore neighborhood are wired together. From these definitions, we can form pixellated antenna patterns by simply redefining the bitstreams of the implied conductivity matrices, yielding for example bowtie (Figure 14b) or spiral antenna (Figure 14c) patterns based on two different bitstreams.

![Figure 14: Example of a Pixellated Antenna](image)

As we have shown in these two examples, it is possible to define a number of different cell types, and we in fact could enumerate non-exhaustively a few other types:

- $σ$-cells, a family of cells relating to the exploitation of conductivity;
- $L$-cells, a family of the familiar lumped element electrical blocks from simple circuit theory (resistors, capacitors, transistors);
- $c$-cells, computation / storage elements (variations of those found in FPGAs);
- $k$-cells, programmable thermal elements;
- $ε$-cells, programmable permittivity;
- $E$-cells, pertaining to energy storage;
- $g$-cells, programmable mechanical attachment;
- $s$-cells, for sensing phenomenologies and converting them to electrical signals;
- $μ$-cells, for programming permeability;
- $η$-cells, for photonic (light-routing) applications;
- $α$-cells, for transduction / actuation (conversion of electrical signals to some other phenomenology, including locomotion);

Most useful at least in conceptual development is the physically unrealizable “universal cell” (the $U$-cell), which can emulate any of the other cells, and possibly do so in parallel. Such a cell would thus carry “one of everything” and we could imagine “synthesizing” a spacecraft (or other cellular system) as an arrangement of $U$-cells initially, which through iterative refinement, might be distilled into a set of practical arrangements of cell types drawn from a finite library of components. Hence, it is highly likely that the cells used for CellSat systems will be heterogeneous, a mixture of different numbers of different cell types, whose distributions will likely depend on mission or purpose.
In these discussions, we have tended to examine cases involving very primitive elements. At the level of a spacecraft, however, we may find it more useful to consider modular blocks having a higher level of sophistication (and probably size) rather than to focus on individual pixels or voxels of elementary function. For example, we might wish to consider a gyro as an s-cell, reaction wheels as α-cells, etc. In this case cells might resemble the blocks of plug-and-play networks, in which components are self-describing and networks self-organizing. Consider the example plug-and-play network in Figure 15. In this diagram, components of a typical satellite are represented as “black boxes” where hubs or routers are employed to achieve the simple connected topologies required to span the network.

![Figure 15: Example PnP Network](image)

We might propose a cellular-inspired version of the same network, as shown in Figure 16, in which case we consider again the \( r=1 \) 2-D Moore neighborhood, but do not show a rigid matrix but suggest a loose, even free-form arrangement of elements. In this case, routers are integrated into all nodes, invoked as needed to form the network of the plug-and-play system.

![Figure 16: Cellular-inspired PnP Network](image)

**Push-button toolflow.** CellSat benefits from “intelligent design.” It is indeed interesting to speculate on the possibilities on Darwinian selection applied in an artificial environment. For example, under appropriate constraints, bitstream variations may be permitted and evaluated (analogous to the approaches of genetic algorithms) for “goodness.” On the scale of sophistication of a spacecraft, using such methods would be quite inefficient based on our present understanding of these types of algorithms.

Until such time as breakthroughs in “evolutionary search”\(^4\) make complex synthesis tractable, we will fall back to heuristic methods, such as the highly automated ones notionally suggested in Figure 17. The rough process flow for designing a CellSat would involve a beginning phase of “problem capture” in which a user translates mission needs into a form that leads to assignments of mass budget, function, and structure. CellSats would be composed through processes similar to those used to compile silicon designs in electronic design automation (EDA) tools. In yet another biological analogy, we would recognize a number of co-existing systems for the creation and distribution of power, signals, and heat, and compile those networks into the eventual specification of a collection of cells (drawn from a “standard cell library”), their arrangements, and bitstreams. The compilation itself produces a string-like representation that is analogous to the DNA of a completely-specified satellite. We might call it a “satellite mashup,” a compound-complex set of individual binary data, meaningful only when the pieces are present and in an arrangement encoded within the mashup itself. The mashup could be extended to include an executable representation of the console used to operate the CellSat, perhaps even mobile code forms suitable for execution on user equipment (like CellSat “plug-ins”).

![Figure 17: Pushbutton Toolflow--Automated CellSat Design and Configuration](image)

---

\(^4\) We credit the term “evolutionary search” to Jason Lohn, who has demonstrated some very intriguing work of this type in design of, for example, electromagnetic components [20].
**CellSat Self-assembly.** The “DNA” produced by the push-button toolflow provides the instructions for components (types required and quantity) of the CellSat, along with their arrangement and detailed configuration. How far do we carry the mimicry of nature? Do we expect, as in nature, that CellSats would actually assemble themselves? We should consider briefly the implications of self-assembly and the circumstances where it would make sense.

In the simplest / easiest case, the mashup is not directly executable by the pieces of a CellSat but require full human / robotic intervention, more akin to the blueprint-like specification of a traditional system. No cues are provided by the system or its pieces, even though the mashup is automatically generated and is arbitrarily complex and complete. In this case, the cells neither self-assemble nor self-organize. This allows minimal overhead, but leads to systems that may not be as robust.

The next level of automation, perhaps, is similar to our previously-described guided self-assembly or variations on that theme. At this level, the components can contribute to their own construction electrically, but cannot actuate their own self-assembly—they require human / robotic intervention. Do we, as in the case of non-trivial organisms, expect cells to carry full copies of their own DNA in every cell? In part, the level of informatic redundancy is governed by architecture choice and convention. In the simpler forms of natural self-assembly (i.e., simple viruses), only one copy of the assembly program exists within the cell. The protein sub-units implement a rule set, and the high symmetry of the capsids leads to a reasonably compact rule set. For systems we traditionally build, the blueprints are spelled out in great detail. In cellular automata, we see obvious cases where a simple local rule allows the expression of reasonably complex structures. Consider the 2-D example in Figure 18 (which is actually a time-unrolled 1-D example) of a cellular automata using a simple XOR rule (a cell is black if an odd number of the cells on the preceding row immediately left, above, and right of that cell are black). In a CellSat, such a simple rule can encode a fractal structure far more compactly than an arbitrary structure created with no notions of representational economy.

It is likely that even when many such local rules can be found, there will be much “non-compressible” information content in the specification of a CellSat. Protocols, even expressive languages, may need to be invented to lead to efficient representations of such cellular systems.

In “full-blown” self-assembly, it is necessary to add the ability for cells to support programmable mechanical attachment, and possibly locomotion. It is whimsically possible to consider a passive self-assembly in which the requisite elements of a CellSat are placed in a sort of fluid suspension and subject to agitation (like a blender) in which opportunistic motion supplies the necessary kinetics for self-assembly and elements simply lock into place when the right piecewise arrangements are encountered. In the most ambitious form of self-assembly, cells might employ their own locomotion, in the form of smart cilia (in which pieces crawl past each other) or three-space locomotion in solution or in vacuo. In either case, the notion of an assembly transition matrix applies, in which the detailed sequence of steps is encoded. Some researchers have explored the representation of artificial self-assembly using graph grammars [4] in which pieces and partial assemblies are encoded as states or nodes in a graph structure, and edges between states represent legal moves, weighted possibly by some notion of probability dictated by kinetics or circumstance. In nature, we see intriguing concepts in complex self-assembly, such as the existence of scaffolding structures. Scaffolding plays an important role in complex assembly by providing transition structures that facilitate (like mandrels) the erection of more sophisticated structures. The scaffolds are created through self-assembly, and dissolved when no longer needed. Transition matrix concepts (such as graph grammars, and likely finite automata and regular expression syntax) appear to support the necessary semantic richness to express even the most complex sequences leading the sophistication of systems such as those envisioned in CellSat. These same protocols could undoubtedly be used to permit the innate self-healing and self-replication of CellSats. This, too, would be implemented in the “mashup” description.

We would envision that as the granularity of CellSats diminish through perhaps a Moore’s [21] Law-like principle, the need for self-assembly would be more evident. Even at “golf-ball” size, the scale of integration is over $10^5$ cells/m$^3$, and for 1mm cells, the scale is $10^9$ cells/m$^3$. It seems unlikely that any cellular
CONCLUSION

In this paper, we have considered a concept for a biologically-inspired architecture to create complex systems (such as spacecraft) based on a cellular approach. In this “cellularity motif” a CellSat would be created as a configurable ensemble of configurable cells, each capable of performing one or more primitive functions, each capable of interacting with neighboring cells through programmable rule systems. In a DNA-like analogy, we could specify configurations as a large binary string, automatically generated by a powerful computer-aided toolflow. We would start our development modestly, in the spirit of a disruptive technology, with simple versions of cell architectures, using cells that are physically large, eventually, through a Moore’s Law-like principle, scaling the size of cells smaller and more numerous to some infinitesimal-size limit.

Like “real” biological systems, the CellSat could heal when injured, reproduce, and mutate. The rule infrastructure provides for these possibilities. Unfortunately, we can also expect to find artificial equivalents of infections and cancer, in which “bad” cells or corrupted DNA can have system-wide impacts.

While this vision seems whimsical at one level, many of the technologies needed are very well-established. Building the first “CellSats” will probably be simpler than understanding them, to find the patterns and languages to best exploit them, and to discover and contain the pathologies that could affect them. The promise is great, a new kind of technology-driven final frontier, in which systems morph before our eyes, from bins of seemingly inert material, into the systems of tomorrow.

ACKNOWLEDGMENTS

This paper was a creative stimulus, stemming most recently from the AFOSR-sponsored workshop on reconfigurable systems, held 14-15 February 2007 in Albuquerque, NM. Work was partially supported by AFOSR’s “Discovery Challenge Thrust” in reconfigurable cellular and electronic photonic arrays.

REFERENCES

15. Proceedings of the 2000 DARPA Moletronics Principal Investigator’s Meeting, South Lake Tahoe, CA.

16. Proceedings of the 2001 DARPA Defense Science Office Molecular Electronics Principal Investigator’s Meeting, Santa Fe, NM.


