Building Critical Systems as a Cyborg

Greg Ball
cyborgg.com

In science fiction, a cyborg is a marriage of machine and human flesh. I'm not suggesting that you turn your favorite officer into an espresso machine, but this article explains the seemingly outrageous possibility that cybernetics may be the next step in the evolution of critical systems, demonstrates actual code and technology that is available, and describes real-world experiences in using it. The strength of this technology is in its resilience and adaptability in building complex critical systems that must face the real world. However, its use requires a shift in thinking about software—much like the introduction of "object-oriented" concepts once did.

Cybernetics is the study of communication and control processes, especially the comparison of these processes in biological and artificial systems. It attempts to learn principles that can be applied to any type of system regardless of its material realization. This kind of study began long before the existence of the modern digital computer. The term itself goes back to Plato.

Don't assume those early cyberneticists would be impressed by our modern high-availability computer systems. They might even view our conventional approach to software as fatally arrogant, requiring a programmer to anticipate everything.

Conventional software is based on the algorithmic approach pioneered by John von Neumann in the 1940s. An algorithm is just "a series of steps to achieve a desired aim" [1] that we then give to our machines to execute. It is a well-behaved approach with predictable results—so long as all of your assumptions are valid, your code is perfect, the world doesn't change, and your enemies are powerless to interfere.

I assume you've experienced what happens otherwise. The more critical a conventional system is, the more rigidly and exhaustively we must define those steps. We must also carefully control its runtime environment. According to the highest standards of compliance (e.g., DO-178B/ED-12B or MIL-STD-498) we must test every possible decision, every pathway, and every conceivable combination of data.

If certification is required, then the cost to produce the associated verification evidence grows exponentially with the size of the application. At some point, this is impossible—even in a modestly complex closed system. And in an open system, we can't even control the scope of the problem.

I sometimes wonder if, like an overprotective parent, our emphasis on rigor hasn't actually made our systems more vulnerable. Whenever our conventional systems encounter something other than the sterile environment that we intended, what sort of coping skills have we given them?

Von Neumann himself wrote about an alternative neural approach, one in which new behaviors can emerge in response to changes in the environment. This would fit the theoretical principles of our cyberneticists exactly, as they emphasize the use of feedback to accomplish goals rather than following a predetermined set of steps. While a neural or cybernetic approach is less well-behaved and less predictable than the software we are used to, it is also extremely adaptable and powerful.

Rather than spending too much time on a soapbox, I would rather present you with a question: Given the right tools, could you design a system that is safer and more economical to build because it has the ability to overcome its own imperfections and environmental obstacles and still complete the mission? Assuming that you are at least thinking about it, let's talk about how you might go about designing such a system.

What Is a Cyborg?

We want both kinds of the behavior that I've talked about, with predictable systems that follow established rules and procedures. But we also want them to adapt in the face of the unexpected. So it would seem that what we need is a hybrid approach: a combination of cybernetics technology with some other type of system. And that's a fairly good working definition of a cyborg. Fair, but not great; it is a bit like describing a car as "something with tires."

The original authoritative definition was published by Dr. Nathan S. Kline and Manfred Clynes in the September 1960 issue of the scientific journal Astronautics. And yes, they did suggest that the bodies of pilots could be modified for space travel using drugs and assorted parts (yikes, can't imagine why that wasn't popular). But those sensational examples were not part of the definition. Instead, they proposed a cybernetic principle that can be applied to any type of system. In their own words:

> What are some of the devices necessary for creating self-regulating man-machine systems? This self-regulation must function without the benefit of consciousness in order to cooperate with the body's own autonomous homeostatic controls. For the exogenously extended organizational complex functioning as an integrated homeostatic system unconsciously, we propose the term "Cyborg." [2]

Homeostatic is the idea of an open system that can regulate itself to function effectively in a broad range of conditions.

Open, as used here, refers to a system in which energy or material (resources) can be added or lost. It also means that the type and number of parts that make up the system are not static.

Exogenous in this context means any material that is present and active in an individual organism but that originated outside of that organism. It is meant to describe a cyborg's blended nature, where control is extended over other non-cybernetic parts.

A cyborg has the authority to unconsciously alter its operation. This language coincides with their example of the human autonomic nervous system. For example, you don't generally think about breathing. You *can* control it, but normally you concentrate on the mission while the body adjusts to your activities, environmental conditions, threats, etc.

A cyborg may alter its operation, but only to maintain a stable state or accomplish goals that we've set for it. Therefore, this definition both empowers and sets specific limits on the authority that is given to a cyborg.

One thing that the original definition does not explicitly mention is the concept of self—though you might infer that from the root words *cybernetic organism*: An organism is a separate distinct individual.

In my opinion, a cyborg must be able to distinguish *self* from any other organism, or the environment, because it must attempt to regulate only itself.

It must not get confused and try to impose its goals on others. It must not attempt to change or take over the universe. It has to have a clear idea of which parts belong to it and which do not. It must have healthy boundaries to protect itself and play well with others.

Since we're now somewhat stretching the original definition of a cyborg, the term we propose to use is *cyborg gratia* or "for the sake of the cyborg." It means that the cybernetic organism is operating for itself as an independent organism inside a larger social structure.

Social governance is the final piece necessary to complete the concept. In an open system, you have to expect communication and cooperation, but also sometimes conflict between organisms. It is highly desirable to design a resilient system as an ecology of independent, cooperative, and adaptive organisms—one that can embody complex relationships with security and selective trust.

Such systems can align themselves with the changing and varied relationships between partners, alliances, and customers. They also enable a different paradigm for development and maintenance that embraces change and diversity.

Technology in Action

The question now is how to make that a reality. Cyborgg (pronounced "cyborg gee") is commercial open-source cybernetic technology in its second generation. It is impossible to describe everything in a short article—and difficult to know where to start. But I can show that working with a cyborg is not onerous.

Cyborgg employs a heterogeneous network of several neuron types to facilitate the integration of these cybernetic extensions into the rest of your system[1]. The data that they work with is not limited to numerical values[2]. They fall into two general classifications:

- Afferent (or sensory) neurons are used to receive input.
- Efferent neurons are used to manipulate or interact with the outside world.

One reason cyborgg was made open was to drive consensus on some basic terms and standards. For example: Just as the format of an e-mail address is important to everyone, so is standardizing the format of a Cyborg URI[3] (or CURI).

CURIs are a key mechanism for surgically implanting complex cybernetic components in conventional code, as shown in the following three examples:

EXAMPLE No. 1
Hooking into the neural net: We provide it with some feedback on system performance using a cyborg helper class.

```
// Define the neuron in question
CURI curi = new
CURI("curi://afferent/response
time$my service");

// You could obtain and use the neuron directly
// but this helper class is convenient
TimeMarker marker =
TimeMarker.start(curi);

// Do something that you want to measure, then
marker.stopAndRecordTime();
```

EXAMPLE No. 2
Another indirect way of hooking into the neural net includes defining an attribute for a class that cyborgg will dynamically control. The following organelle shown is a convenient wrapper around a neuron that implies that this class is an organ. But where is the CURI? Cyborgg creates it behind the scenes by inspecting the rest of the class:

```
protected Organelle queueSize =
Organelle.newInteger("queue size",
10);
```

EXAMPLE No. 3
One can also select a cybernetic component or service. Note that many factors are in play here including failover, system load, authentication, biases, automated service discovery, etc. But these are handled invisibly by the cybernetic core:

```
// Asking for a certain type of service with no filters or restrictions
CURI curi = new CURI("curi://$my service");

// Cyborgg class used for dynamic dependency injection
Injector injector  = new Injector();
// obtain the service
MyService service = injector.use(curi);
```

What happens when you do something like this? Under the hood, cyborgg complies with Aleksander's definition of a neural net as having a "network of adaptable nodes which, through a process of learning, store experiential knowledge and make it available for use" [3][4]. New system behaviors can and do arise from changes to the structure of this net.

Each cyborgg neuron has a complex internal structure that is more cell-like than the classical neural approaches[5]. This was done to overcome two barriers to general use.

The first barrier was the difficulty of understanding and having confidence in the decisions made by the net (analysis of a classical neural net is something only a researcher could love).

To address this, each cyborgg neuron contains a nucleus—a statistical model that captures information about the neuron's behavior (most of the information that a Six Sigma practitioner would ask for). Therefore, when a neuron fires, it can tell you plainly things like "with 80 percent certainty, these adjustments to the system are predicted to change the behavior of a certain aspect by 68 percent (plus or minus 10 percent)."

You can track the difference between the neuron's prediction and the actual result—and track the performance of the system in general. It keeps the cyborg from acting on weak or invalid assumptions. It is also used to discover new or unexpected correlations. This not only gives the cyborg power, but allows it to serve as a research tool.

The second barrier was the difficulty of training the net, of knowing what synapses to forge, and how the health of one neuron is related to the health of another. So each neuron contains a genetic algorithm or genotype that is used to grow and test new relationships. A pluggable axon allows the system to grapple with how it should respond[6]. Both of these are guided during the cyborg configuration.

This configuration includes the ability to define rules to shape, as well as extend or modify, the cyborg's behavior even after the system is deployed and running. It will obey broadcasted commands (using an extensible lexical command processor), including built-in diagnostic and test commands.

I have barely scratched the surface of just this one aspect, and there is no room to explain how service selection and failover takes place. There so much to describe about the social structure that is a central part of that decision—or the communication and other technology that supports it. Still, I've made clear what the purposes of a cyborg are and presented enough about the technology to encourage you to explore and test it for yourselves.

Lessons of Use

Cyborgg is currently in use supporting a number of medical facilities from cancer research to small practices. It is the technical foundation or glue inside a leading health care software vendor's product, which brings together a distributed group of components from multiple vendors into a single enterprise services bus for health care. Its application has included electronic medical record services, disease management, clinical trials management, transcription, and document scanning.

Among the lessons learned in its use (so far):

- Configuration of a large distributed adaptable system can be problematic, which led to a redesign of the configuration subsystem in the latest version of cyborgg[7].
- The amount of benefit that you receive is closely related to the way that you modularize and package your system. Greater benefits come when applications are not monolithic.
- We found that good modular designs were sometimes negated by the deployment model. This led to the introduction of organs as an additional cyborgg concept, and Java Network Launch Protocol Replaceable Units as a supporting service or technology.
- The concepts are currently different and new enough to require good training of your development team. It particularly rewards a savvy architect that takes the time to learn its capabilities.
- Visibility is a key organizational success factor, underlining the importance of features to allow technical users to interact with a cyborg.

The strength of this technology is in building complex critical systems that defense organizations face in the real world. That the old combat phrase "no plan survives contact with the enemy" still holds true, as is the belief that any system that cannot adapt is likely to fail. Cyborg technology represents a controlled step defense organizations can take to be more adaptable—away from their stiff, pre-programmed conventional software and towards systems that have greater problem-solving skills.◆

References

1. von Neumann, John, Arthur Burks, and Herman H. Goldstine. *Preliminary Discussion of the Logical Design of an Electronic Computing Instrument*. U.S. Army Ordinance Department Report. June 1946.
2. Clynes, Manfred E., and Nathan S. Kline. "Cyborgs and Space." *Astronautics* (Sept. 1960): 26-27 and 74-75.
3. Aleksander, Igor, and Helen Morton. *An Introduction to Neural Computing*. London: International Thomson Computer, 1995.

Notes

1. For example, motor neurons work with threaded processes, failure analysis neurons are for failure analysis, and *germ layers* work with exogenous services in a service-oriented architecture.
2. Strings and other non-numeric data are converted to ordinals.
3. A Uniform Resource Identifier (URI) is like the familiar Uniform Resource Locator (URL) except that the resource it identifies does not necessarily specify location. A URL is a type of URI.
4. Igor Aleksander's work led to the development of the first computer based on neural principles to reach the marketplace.
5. For example, Boltzmann machines, Kohonen maps, and perceptrons.
6. Slipping into the math for a minute, the relationship between real-world neurons is likely not a simple linear relationship. Ask questions such as "Should it be quadratic?" and "Should we use a radial basis function?" The default axon uses a type of ragged cube that allows each neuron to fire according to a data-driven complex curve. However, the cyborgg application programming interface allows and encourages the researcher to substitute their own firing function and measure its effectiveness versus other approaches.
7. Changes included: automatic discovery of cyborgg-enabled services; the option to use configuration references where a commonly used set of goals or other parameters is defined only once, using a unique identification, and then referenced by other components; and with the addition of new installation and configuration services that allow the pushing of upgrades to remote customers.

CMMI Level 5 and Rising
309th Software Maintenance Group
Ogden Air Logistics Center
309th Software Maintenance Group
Hill Air Force Base, UT 84056
801-777-2615
DSN: 777-2615
www.309smxg.hill.af.mil
1
2
3
4
5
Excellent Results
Fully Satisfied Customers

About the Author

Greg Ball is a software and system architect with more than 15 years of government and private industry experience. He prefers a complex technical problem to most other forms of entertainment. Ball is the original creator of cyborgg.

7911 Woodstone LN
Dallas, TX 75248
E-mail: gball@cyborgg.com