Early in WWI, zeppelin airships were used by Germany to bomb London and Paris. They were not very successful, due to the obvious vulnerability presented by their large and fragile bags of highly explosive gas floating gently above a war zone. If we think of what it would take to shoot down one of these devices using surface artillery, it becomes obvious that success is determined by accuracy of measurement.

To successfully bag a zeppelin, we would need to know, as precisely as possible, the following information: for the airship—altitude, distance down range, velocity, size; for the ground cannon, we'd need to know muzzle velocity, projectile air resistance, projectile weight, and so forth; for the atmosphere, we'd need to know air resistance, temperature/density, wind velocity, among others. The mathematical equation that relates these variables is rather complex, and some of the first-ever computer applications were developed to compute similar functions. However, our functional ability to destroy the threat (assuming we have a suitably powerful cannon) is dependent on our ability to (1) collect this information as accurately as possible, (2) relay it to the gun, and (3) apply it to the aiming of the weapon. The information must be available just at the point of firing, and our success is almost entirely dependent upon the accuracy with which we can accomplish these things.

Shooting Down Supersonic Planes

Today, we don't shoot down zeppelins; we shoot down fast, low-flying supersonic jet planes. There are a lot of differences. First, we would probably not use ballistic ordnance at all to hit a low-flying, fast plane. If we did, we would certainly not shoot just one projectile, we would fill the sky with the stuff, hoping for a lucky hit. We would more likely use a ground- or air-launched guided missile of some sort. One of the major differences between scenarios is that in the case of the zeppelin, we can predict the intersection of the airship and the antiaircraft projectile. In the case of the jet, not only do we not know in advance exactly where in space the missile will intercept the plane, we don't need to know. The reason why we cannot predict the point of impact is, of course, that the plane will likely maneuver to escape the missile.

Zeppelins and Jet Planes: A Metaphor for Modern Software Projects

Comparing modern-day project characteristics with yesteryear's.
The fact that the missile is trying to hit the plane causes the plane to, unpredictably, change direction.

Here is the metaphor: the zeppelin scenario represents projects of yesteryear, 20 or more years ago; the jet plane scenario represents today’s projects. If we look at the characteristics of each, we can start to see some similarities:

- **Zeppelin**: slow-moving, large, and well-defined target (compared to the speed of the projectile), deterministic and predictable approach, predictable response by target within range of operation of targeting system, success defined by advance knowledge and precision.
- **Jet Plane**: fast-moving, small target, may be employing countermeasures, unpredictable response, target speed close to projectile speed, success determined by flexibility of response.

Yesterday’s zeppelin projects were the classic mainframe business systems and even, in some respects, classical engineering systems. They were typified by well-defined, often static, goals. The rate of change of the target system was usually well within the normal flexibility of the project. Time frames were often quite long, and the projects dealt with (relatively) few variables. The progress was predictable and linear—requirements design, code, test, bang! Such projects could be highly organized with detailed and effective processes, controlled from a single point, and could effectively employ single tasking activities.

Today’s jet fighter projects are very different. The target appears and disappears across an increasingly narrow window of opportunity. We have to launch the project, sometimes without a certainty of a hit. We launch, not on an exact estimate, but on the best probability of acquiring the target, and we have to accept probabilities of less than 100%. The project may run away from us as fast as we can track it. It will certainly change direction and progress may be cyclical.

The very act of firing (attempting to build the system) will cause the target to evade (the requirements to change). This is an intrinsically nondeterministic situation that cannot be effectively controlled entirely from the “ground.” The projects must not only be a heck of a lot faster than they used to be, they also must be much more flexible.

This implies single tasking and highly controlled projects cannot be successful.

**The Zeppelin Legacy**

The renaissance period of software development methods was between about 1975 and say 1990. Not that there haven’t been great strides since then, but this period saw the first approaches to establishing methods, process, and management models for software production. Much of what we view as modern methods: object orientation, systems engineering, life cycle management, many programming languages, inspections, the Internet, GUIs, and so on, were originally devised back then (most of them, it seems, at Xerox’s PARC research center, which was as famous for inventing things as it was for being unable to make any money from them). If we look at the predominantly large systems being built at the time, they were usually of the zeppelin type. The trouble with this is several-fold: (1) being zeppelin systems, they needed and caused the creation of anti-zeppelin processes; (2) these processes, being highly deterministic, fed the idea that software is a product to be manufactured (which it was then, in fact); (3) subsequent processes, methods, and languages were built on the foundation and expectation of predictability, which feeds nicely into most companies’ needs to be in control of what is happening and likely to happen; and (4) the developers who grew up in
this era, and now occupy the higher levels of modern corporations, gained a worldview that said this is something that can be controlled. That is, most of our methods, processes, and certainly our management models of this activity were actually designed for something quite different, in a much simpler era. We are still paying a price for this, especially in the area of process.

**Supersonic Jet Projects**

It is unfortunate that most of our projects these days are so fraught with change that we cannot “lock down” requirements; customers and the marketplace change their minds so often, we can’t catch up. It is irritating that technology changes faster than we can absorb it. It is painful that we cannot estimate projects accurately in advance. And it is galling that things do not go as we expect them to as managers in charge of projects, or need them to as companies laying down huge financial commitments. But this is the reality. It’s rather like gravity. We might not like it, we might wish it weren’t there, or operated differently. But we still shouldn’t step off a tall building without taking it into account.

To thoroughly abuse the metaphor, here are some equivalences:

- Use missiles, not ballistic ordnance=Use short-cycle, flexible projects.
- Launch when there’s the best chance of a hit=Use probabilistic estimation techniques and actively assess and manage risk.
- Use self-guided missiles=Build autonomous, self-directed work teams.
- Adjust trajectories as needed.
- Use real-time target acquisition systems=Develop intelligent data collection methods and apply them in real time to the project.
- Employ counter-countermeasures=Anticipate and manage market and customer reaction.
- Build the next generation weapons systems=Stay ahead technologically.

Probably the most significant, and the most difficult change for most organizations is this: give up the need for control. We are in a situation where, at least to some degree, we cannot control the outcome. This is not so much a problem, unless explicit control is one of our requirements. The way out of a situation that cannot be controlled is not more control. It is to develop our capability—our project teams, and the systems that support them. And then to turn over the keys to them in the sure, but uncomfortable, hope they can navigate to the target.

Phil Armour (armour@corvusintl.com) is a vice president and senior consultant at Corvus International Inc., Deer Park, IL.