Software Surprise
The Three Invisible Problems of Weapons System Software Development

Lt. Col. Lionel D. Alford Jr.
U.S. Air Force

This article describes how software-induced workload, software system complexity, and software systems cost may cause rarely identified but long-lasting adverse effects to a program. If you cannot find these three problems in your software development program, you may not realize what hit you until it is too late.

Software in developmental systems causes three critical challenges for the manager of a system program office (SPO), the testers, and ultimately the customers: software-induced workload, software system complexity, and software systems cost. For each of these challenges, you could insert the word "integrated" in place of "software"; the result is the same. Even though these three problems have an enormous effect on the overall system, they are given little attention because SPOs rarely realize they exist. All three of these problems are "program invisible"—they are rarely tested or given any thought until after they have become a serious difficulty for the SPO. The problem is that these software and integration problems are some of the foremost reasons for customer dissatisfaction and increased systems cost.

Software-Induced Workload

SPOs attempt to reduce software-induced workload by adding software to the system. Current hardware and the missions this hardware supports are extremely complex. Software is primarily used to integrate and consolidate complex systems so the equipment operators can accomplish the mission with decreased workload and increased mission effectiveness. However, no one has discovered a way to measure workload. All the measures we currently have for workload are qualitative and not quantitative.

In the past, engineers tried to use quantitative measures such as altitude and airspeed capture to measure workload; unfortunately, these measures have nothing to do with workload. Take the example of a test pilot who is required to use digital instruments to keep an aircraft within 10 feet above or below a target altitude. According to conventional engineering measures, the workload should not be great because the event falls within the realm of possibility; nevertheless, the workload is extremely high—the pilot has to constantly work the controls and interpret instruments. Even a test pilot cannot accomplish this task for long. After a series of engineering analyses, tape altimeters were installed on the C-5, C-141, F-111, and FB-111 aircraft. Aviators who have flown these aircraft will testify to their "low" workload after they have become proficient in the systems; however, controlled analytical tests with other aviators using standard instruments always show that their perceptions are inaccurate.

Because there is no usable measure for workload, when we try to measure workload, data from such analyses are always suspect: The sample size is rarely large, the statistical confidence is low, and there is no method to quantitatively measure the workload. Since we use these analyses when evaluating whether we want to reduce the number of crew members in the cockpit, for instance, it is not a decision based on analysis and test; it is a hope based on politics and the cost of the additional crew members. The best examples of this are the MC-130H aircraft and the current U.S. Air Force glass cockpits and heads-up displays (HUDs). On the one hand, the MC-130H is one of the best missionized aircraft in the world. The pilot puts the cue on the dot and can fly any terrain by following profile programmed by the navigator and the aircraft system. On the other hand, it is a poor instrument aircraft. The tape digital displays make it extremely difficult to fly. In like fashion, the glass cockpits and HUDs of Air Force aircraft are based on similar tape displays. These displays work well for civil aircraft, which are flown from takeoff to touchdown on the autopilot, but they are "workload sinks" for military tactical flight. This workload problem will continue to be an obstacle until a method to quantitatively measure workload is discovered.

Fortunately, there is research toward this end, but a majority of fielded and future systems have been or are being designed without regard to the workload involved. A final example is radio frequency changes in aircraft that use digital integrated radio systems. It is simple to change a frequency using the old analogue dial paradigm—the pilot inputs the frequency by turning a dial on the console. But in a software display, the pilot must first find the page for frequency entry, then select the proper place for the entry, and finally, input the digits from a touch pad. This is at least 10 times greater workload than the analogue dialing system, yet it is the new paradigm. If you multiply the workload in this example by the number of system inputs the pilot must make to accomplish any mission, it will demonstrate only a small fraction of the magnitude of problems associated with workload. It is enough to say that software and integrated systems generally have significantly increased workload without a proportional increase in mission effectiveness.

Software Complexity

The second great hidden problem in software development is software complexity. Because software is so intrusive—that is, it affects many systems—it has become impossible to fully test even the safety-related effects of the software. When a new software build is installed in an aircraft, unknowns are rampant, and the "bugs" are rarely fully discovered during flight test. Some problems lie dormant until the systems are well deployed. One example was an operational flight program (OFP) release on the MC-130H. This release was supposed to affect only the terrain-following (TF) system of the aircraft. The aircraft was released for flight under the assumption that it would operate properly as long as the TF system was not engaged. In the middle of a training flight, during an engine-out approach, the crew noticed that the "ball" (primary flight coordination instrument) was indicating sideslip in the opposite direction. Because the TF system was an integral part of the OFP, a change to the TF system software resulted in an erroneous reading in another part of the system. If this OFP had made it into the fleet, or an experienced test crew had not been flying the aircraft, it is likely there would have been a smoking hole where a multimillion-dollar aircraft once had been. This is an extreme example, but there have been hundreds of others in and out of flight test.

Software and integrated systems increase risk proportional to the increasing code and increasing integration complexity. In the C-21 aircraft (Lear 35), a pulled or popped oil pressure circuit would cause the engine control settings to indicate fire on an engine. An operational crew discovered this problem when they got two fire lights, one on each engine. They had to shut down a good engine and land short of their destination. They were lucky to realize there was a problem with the indicating system before they shut down both engines. The circuit breaker had popped due to a faulty circuit problem, and a sneak circuit caused the fire warning in the indicating system. A $10 piece of equipment gave the software false information, and the crew and passengers were placed at risk because testing had not been done with the oil pressure circuits pulled. This defect has been fixed since the incident, but who knows how many similar problems wait to be found? Software and integration complexity increase risk.

Software Systems Cost

The third problem is related to the first two. Software always requires future improvements and rewrites. Complex software invariably comes with bugs that are never entirely discovered. Modifications and fixes add more bugs, which results in future modifications and fixes. Rarely are software systems provided with sufficient lifecycle funding for these processes.

Software has become so intrusive that the simplest components on many aircraft incorporate some software. In fact, such things as the clocks, circuit breakers, and pressurization systems in most modern aircraft incorporate or are dependent on software for correct indication and operation. Most aircraft are now to some degree fly-by-wire and engine control-by-wire. This trend toward software-driven controls and systems shows no sign of change or reversal. Therefore, funding must be provided for any software system until the decommission of the system—a given that has not been acknowledged by most services and program offices. For example, there are numerous electronic warfare systems that are not adequately funded for software changes but are nevertheless going through major changes. This has resulted in serious program problems such as multiple OFPs in multiple versions being deployed by more than one agency. The resulting costs are much more than they would have been if software changes had been planned for the life of the system. The examples of the MC-130H and the C-21 resulted in unplanned cost increases that could have radically affected the safety of the aircraft if the funding had not been made available.

Conclusion

The lessons to learn from these three invisible software and integration problems are simple—their solutions are not. First, try to evaluate workload when developing a system. Attempt to use nonintegrated systems when possible, especially when workload studies indicate a problem. The Department of Defense must fund research and development to discover effective quantitative workload measures. Second, plan and test for as much as possible, and be ready—during all program phases—for software problems to rear their ugly heads. Do not be content with minimal software testing even when risk is low. Finally, fund software for the life of the system.

These three issues are critical, rarely visible problems. They should be primary considerations during all SPO phases. They may be invisible now, but unless tamed, they will drive your program and the capability of your weapons system. ◆

About the Author

Lt. Col. Lionel D. Alford Jr. is the chief of the Special Operations Forces Test and Evaluation Division, Wright-Patterson Air Force Base, Ohio. He is an Air Force experimental test pilot with over 3,600 hours in more than 40 different type aircraft and is a member of the Society of Experimental Test Pilots. Alford has served as the chief of the Testing Commercial Aircraft for Military Acquisition Office at Edwards Air Force Base, Calif., holds an Airline Transport Pilot license, and was the chief test pilot for a number of Air Force acquisitions. He is a graduate of the Defense Systems Management College Advanced Program Management Course 98-1. He has a master's degree in mechanical engineering from Boston University and a bachelor's degree in chemistry from Pacific Lutheran University.

ASC/LUQ
2275 D Street, Room 142
Wright-Patterson AFB, OH 45433
Voice: 937-255-9311
Fax: 937-255-0995
E-mail: Pilotlion@aol.com