War Upon the Map:
The Politics of Military User Innovation

Jon R. Lindsay
Doctoral Candidate
Massachusetts Institute of Technology
Department of Political Science
DRAFT v3.0
21 June 2006
Word Count: 17,200 (Footnotes: 5,088)

ABSTRACT: While military personnel are often involved in the design of information technology, the literature on military innovation generally assumes defense contractors are the primary producers. Furthermore, general organizational theories of user innovation have only been tested on cases involving corporate employees or private citizens in substantially less regulated environments than military users. This paper examines user innovation theory in a military context through a historical study of the user-led development of *FalconView*, the popular standard for digital mapping applications throughout the U.S. military and some other government organizations. This paper finds that while user innovation theory can explain aspects of the emergence and diffusion of military user innovation, existing theory understates the challenges involved with generating and sustaining user innovation within a complex bureaucracy. Successfully innovating users must be creative with organizational as well as technical resources.

The debate over the effect of information technology on war—often called the Revolution in Military Affairs (RMA)—has attracted enthusiastic forecasts of sweeping change[1] and skeptical criticism,[2] yet it has cast surprisingly little light on the practical day-to-day experience of computer users in military organizations. Too rarely noticed is that a lot of software in use within military organizations is actually designed or modified by personnel, often working around the problems of traditionally procured software programs. User-developed macros, databases, low-level utilities, web pages, and even sophisticated programs can be found on military networks, interwoven with commercially produced software and custom software written by defense-contractors. The iconic RMA image of lightning bolts arcing around a network of ships, aircraft,

[1] John Arquilla and David F. Ronfeldt, *In Athena's Camp: Preparing For Conflict in the Information Age* (Santa Monica, CA: RAND Corporation, 1997); David S. Alberts et al., *Network Centric Warfare: Developing and Leveraging Information Superiority* (Washington D.C.: CCRP Publications Series, 1999); William A. Owens, *Lifting the Fog of War* (New York, NY: Farrar, Straus and Giroux, 2000).

[2] Jeremy Shapiro, "Information and War: Is It a Revolution?" in *Strategic Appraisal: the Changing Role of Information in Warfare*, ed. Zalmay Khalilzad (Santa Monica, CA: RAND Corporation, 1999), 113-153; Michael E. O'Hanlon, *Technological Change and the Future of Warfare* (Washington DC: Brookings Institution Press, 2000); Eliot A. Cohen, "Change and Transformation in Military Affairs," *Journal of Strategic Studies* vol. 27, no. 3 (2004): 395-407; Stephen Biddle, *Military Power: Explaining Victory and Defeat in Modern Battle* (Princeton University Press, 2004).

satellites, and ground vehicles obscures the fact that low-level innovation is oiling the entire command, control, and intelligence machine.[3]

Improvisation is an important part of organizational life,[4] yet is especially pronounced with information technology because of its complex and changeable insinuation into organizational practices.[5] For military organizations as well, improvisations, field expedients, jury-rigs, and technical stratagems abound in combat histories.[6] Ever since the Greeks built the Trojan Horse without bidding the project out to contractors, the frictions and surprises of real operations have exposed shortfalls in doctrine and technology and motivated personnel to cobble together expedient solutions. There are, moreover, several reasons to expect information technology expedients to be especially prevalent. First, military operations have a large and growing information processing load because of sophisticated weapons systems, the intelligence and planning requirements of missions like precision strike and counter-terrorism, and a fluid strategic environment of coalition, interagency, non-governmental, and adversarial actors. Second, the defense acquisition bureaucracy, already inefficient in procuring massive hardware projects, is especially ill adapted to software production.[7] Third, powerful and flexible

[3] As Joshua Davis, "If We Run Out of Batteries, This War Is Screwed," *Wired* (June 2003) reports: "I tracked the network from the generals' plasma screens at Central Command to the forward nodes on the battlefields in Iraq. What I discovered was something entirely different from the shiny picture of techno-supremacy touted by the proponents of the Rumsfeld doctrine. I found an unsung corps of geeks improvising as they went, cobbling together a remarkable system from a hodgepodge of military-built networking technology, off-the-shelf gear, miles of Ethernet cable, and commercial software. And during two weeks in the war zone, I never heard anyone mention the revolution in military affairs."

[4] Wanda J. Orlikowski, "Improvising Organizational Transformation Over Time: A Situated Change Perspective," *Information Systems Research* vol. 7, no. 1 (1996): 63-93, Karl E. Weick, "Introductory Essay: Improvisation As a Mindset For Organizational Analysis," *Organization Science* vol. 9, no. 5 (1998): 543-555, Ted Baker and Reed E. Nelson, "Creating Something From Nothing: Resource Construction Through Entrepreneurial Bricolage," *Administrative Science Quarterly* vol. 50, no. 3 (2005): 329-366.

[5] John Seely Brown and Paul Duguid, *The Social Life of Information* (Boston, MA: Harvard Business School Press, 2000), Claudio Ciborra, *The Labyrinths of Information: Challenging the Wisdom of Systems* (Oxford: Oxford University Press, 2002)

[6] A few texts treat this theme explicitly: John H. Hay, *Tactical and Materiel Innovations, Vietnam Studies* (Washington, D.C.: Department of the Army, 1974), Center of Military History, *Improvisations During the Russian Campaign* (Washington DC: United States Army, 1986), Michael D. Doubler, *Closing With the Enemy: How GIs Fought the War in Europe, 1944-1945* (University Press of Kansas, 1995)

[7] The Defense Science Board, *Report of the Defense Science Board Task Force on Defense Software* (November 2000), found that "software is rapidly becoming a significant, if not the most significant, portion of DoD acquisitions" and yet "programs lacked well thought-out, disciplined program management and/or software development processes." The study furthermore noted that almost all the major problems identified in six major Department of Defense studies on software acquisition since 1987 had not been

software tools are available at relatively low cost, and the U.S. military population is increasingly computer literate and well educated. In sum, contemporary U.S. military operations generate a high demand for information processing power, acquisition institutions are ill suited to efficiently supply it, and at the same time personnel are empowered to try and supply it for themselves.

Nonetheless, would-be user technology developers in the military lack the legitimacy of official bureaucratic acquisition and certification processes, and they are constrained by short tours-of-duty and other war-fighting responsibilities. Although short-term expedients to deal with the exigencies of combat are an expected part of warfare, in the long-term, military technology is generally procured and managed through a complicated system of contractors, government managers, military staff officers, politicians, and lawyers. Given the high potential for jurisdictional conflict with procurement and technology management regimes, the interesting questions are whether and how spontaneous user innovation can possibly give rise to important long-term organizational consequences.

This paper begins with a discussion of bottom-up user innovation in the literature on military innovation and in business-oriented innovation theory. Because it is missing from the former and has not yet been applied to military organizations within the latter, the paper next frames expectations about user innovation in the idiosyncratic military context. These are next examined in a historical case study of *FalconView*, a software application that emerged through user efforts to develop automated planning tools within the U.S. Air National Guard F-16 community, yet which went on to become the popular standard for operational geospatial applications throughout the military services and some other government agencies as well. This case is then generalized in a discussion of how

---

corrected (pp. ES1-ES2, 3). Similarly, U.S. General Accounting Office, *Stronger Management Practices Are Needed to Improve DOD's Software-Intensive Weapon Acquisitions* (GAO-04-393, 2004) observes, "DOD estimates that it spends about 40 percent of its Research, Development, Test, and Evaluation budget on software—$21 billion for fiscal year 2003. Furthermore, DOD and industry experience indicates that about $8 billion (40 percent) of that amount may be spent on reworking software because of quality-related issues….DOD did not have effective and consistent corporate or software processes for software acquisitions," (p. 1). Also see David C. Gompert et al., *Extending the User's Reach: Responsive Networking For Integrated Military Operations* (National Defense University, Center For Technology and National Security Policy, Defense and Technology Paper 24, 2006) for a detailed critique of information technology procurement.

changes in supply and demand side factors can promote user innovation, while organizational politics nevertheless resist shifting to the new equilibrium.

Explaining Military User Innovation

Like defense procurement institutions, major theories of military innovation have been built around large-scale capital-intensive projects such as tanks, aircraft carriers, and nuclear missiles. For such systems, the question of the source of production—user or contractor?—need never come up. When only private contractors or public arsenals have the capacity for heavy industrial production, a clear division of labor between producer and operational user can safely be assumed. Users may be able to perform some post-production modifications to heavy equipment, as when American soldiers in WWII fashioned hedgerow cutters for Sherman tanks from German beach obstacles,[8] or create small-scale prototypes later mass-produced by contractors, as was the case with submarine emergency escape breathing lungs[9] and the first aviation helmets with embedded radios.[10] However, the primary technological functionality associated with significant military change in the industrial era has generally been built on the assembly line, with significant per-unit costs. This is not necessarily so for software development, where programs *for* a PC can be written *on* a PC within operationally relevant timeframes, where perfect copies can be distributed via floppy disk or networks for negligible cost, and where understanding the detailed context of use is especially critical to the design process. Short of industrial mass-production, a clear division of labor between developer and user cannot be assumed *ex ante*.

The types of theories advanced to explain military innovation include civilian intervention in response to strategic threat,[11] intraservice generational change led by visionary senior officers,[12] and interservice bureaucratic competition.[13] While

[8] Doubler, *Closing With the Enemy*, 44-46.
[9] Peter Maas, *The Terrible Hours: The Man Behind the Greatest Submarine Rescue in History* (New York, NY: HarperCollins Publishers, 1999)
[10] Timothy Scott Wolters, *Managing a Sea of Information: Shipboard Command and Control in the United States Navy, 1899-1945* (Ph.D. Dissertation, MIT Program Science, Technology, and Society, 2003), 102.
[11] Barry R. Posen, *Sources of Military Doctrine: France, Britain and Germany Between the World Wars* (Ithaca NY: Cornell University Press, 1984), Deborah D. Avant, *Political Institutions and Military Change: Lessons From Peripheral Wars* (Cornell University Press, 1994).
[12] Stephen Peter Rosen, *Winning the Next War: Innovation and the Modern Military* (Ithaca, NY: Cornell University Press, 1991)

appreciating that the technology involved in these studies is invariably industrially produced, it's also important to note that these theories are crafted to explain organizational and doctrinal innovations for the employment of technology, rather than the emergence of innovative technology itself. A related literature on the diffusion of military innovation examines diffusion across militaries,[14] but by focusing on industrially produced technology, it also leaves open the question of the functional origin of innovation (user or contractor). As a group these macro-level mechanisms don't speak to whether and how sources of significant technological variation might emerge bottom-up from deep within the organization. While these approaches may become important should user innovation become disruptive enough to require protection or championing from senior leadership, it's nevertheless conceivable that a great deal of change might be initiated and sustained "below the radar" before that occurs.

Some studies of innovation in tactical doctrine highlight the importance of bottom-up trial-and-error and diffusion of successful practices during wartime.[15] The high intensity combat characteristic of the World Wars gave rise to a more-or-less Darwinian process in which many novel tactical variations could be attempted, most perishing with the units that tried them, yet some selected and reinforced through employment and ad-hoc training. These studies focus on tactical employment more than technological design, but the same operational incentives that encourage tactical innovation by personnel could encourage technological innovation as well if barriers to entry for technological design were low enough. Eliot Cohen asserts that "change tends to come more from below, from the spontaneous interactions between military people, technology and particular tactical circumstances,"[16] yet he leaves the task of fleshing out these "spontaneous interactions" to others.

---

[13] Harvey M. Sapolsky, *The Polaris System Development: Bureaucratic and Programmatic Success in Government* (Harvard University Press, 1972), Owen R. Coté, *The Politics of Innovative Military Doctrine: the US Navy and Fleet Ballistic Missiles* (Ph.D. Dissertation, MIT Department of Political Science, 1996)

[14] Thomas W. Zarzecki, *Arms Diffusion: the Spread of Military Innovations in the International System* (New York, NY: Routledge, 2002), Emily O. Goldman and Leslie C. Eliason, *The Diffusion of Military Technology and Ideas* (Palo Alto, CA: Stanford University Press, 2003)

[15] Timothy T. Lupfer, *The Dynamics of Doctrine: The Change in German Tactical Doctrine During the First World War* (Fort Leavenworth, KS: U.S. Army Command and General Staff College, 1981), Doubler, *Closing With the Enemy.*

[16] Cohen, "Change and Transformation in Military Affairs," 400-401.

In the commercial world, users often exploit technology's "interpretive flexibility," sometimes leading to significant industrial change,[17] so organization theory on user innovation, most explicitly articulated by Eric Von Hippel,[18] is a sensible place to begin. A primary hypothesis of Von Hippel's theory is that variation in the functional source of innovation—user, manufacturer, or supplier—is caused by variation in actors' expected innovation-related gains. Users gain from immediate in-house results, manufacturers can gain only when a user market clearly exists, and suppliers gain when they can enhance or complement the market for the goods they supply. Actors' expected gains are influenced by their relative abilities to maintain monopoly control over an innovation, anticipated costs and benefits of the innovation, and transaction costs of alternative contracting arrangements. Transaction costs between users and manufacturers can be quite high because of principal-agent monitoring costs, production time lags, and particularly because knowledge of user needs and manufacturer technologies is often tacit, embodied, and situated,[19] making it difficult to specify accurate requirements. A primary prediction of the theory is that increasing transaction costs while lowering production costs (through changes in user knowledge and resource availability) will shift the functional locus of innovation from manufacturers to users.

The theory advances three further hypotheses about user innovation specifically. First, user innovation should be most prevalent among "lead users," who are users that are experiencing emerging market needs in advance of other users and who stand to immediately benefit from finding a solution. Second, users will tend to freely reveal their innovations to other users because those others may contribute knowledge or resources to improve the innovation, and because free revealing enhances the innovator's reputation and feelings of group solidarity. Third, user innovation tends to be widely distributed in self-reinforcing "innovation communities," where free revealing leads to increasing

[17] Ronald Kline and Trevor Pinch, "Users As Agents of Technological Change: the Social Construction of the Automobile in the Rural United States," *Technology and Culture* vol. 37, no. 4 (1996): 763-795, David N. Lucsko, *Manufacturing Muscle: the Hot Rod Industry and the American Fascination With Speed, 1915-1985* (Ph.D. Dissertation, MIT Program in Science, Technology, and Society, 2005).

[18] Eric Von Hippel, *The Sources of Innovation* (Oxford University Press, 1988), Eric Von Hippel, *Democratizing Innovation* (Cambridge, MA: MIT Press, 2005).

[19] Michael Polanyi, *Personal Knowledge: Towards a Post-Critical Philosophy* (Chicago IL: University of Chicago Press, 1974), Andy Clark, *Being There: Putting Brain, Body, and World Together Again* (Cambridge, MA: MIT Press, 1997), Wanda J. Orlikowski, "Knowing in Practice: Enacting a Collective Capability in Distributed Organizing," *Organization Science* vol. 13, no. 3 (2002): 249-273.

returns for all. This phenomenon is often noted regarding open-source software projects, where a large distributed user group can efficiently identify bugs, find programmers who know how to fix them, and explore idiosyncratic new use cases more efficiently than traditionally managed projects.[20]

Finally, the theory expects production to shift from users to manufacturers once transaction costs diminish and the needs of the user population become well understood, since this shifts the relative expected gain to manufacturers and thus the functional locus of production.[21] As a result, lead users often innovate novel prototypes in emerging markets while manufacturers innovate improvements on established products with healthy markets. Manufacturers can take advantage of this asymmetry by intentionally lowering the cost of user innovation by providing toolkits, such as application programming interfaces (APIs), to facilitate user modification, extension, or design. Of course, if transaction costs don't diminish even after markets are established, then user innovation communities may maintain an enduring competitive advantage (as may be the case with some open-source software projects, since software contracting problems are notoriously difficult).

These hypotheses are well supported by evidence from a wide variety of industries ranging from scientific instrumentation, extreme sports equipment, semiconductor manufacturing, computer hardware, and computer software.[22] However, the users in this sample are exclusively corporate employees or private citizens. Does the theory still work in a military environment, where users are involved in the peculiar business of war and are situated within vast complex bureaucracies?

Unlike any other formal organization, militaries are concerned with the direct, physical destruction of like kinds. Despite their awesome capabilities, military organizations can actually be somewhat fragile because complex uncertainty becomes an existential problem. The stabilizing routines and structures that emerge in response can

[20] Eric S. Raymond, *The Cathedral and the Bazaar (Revised Edition)* (Cambridge, MA: O'Reilly, 2001), Josh Lerner and Jean Tirole, "Some Simple Economics of Open Source," *The Journal of Industrial Economics* vol. 50, no. 2 (2002): 197-234

[21] Carliss Baldwin et al., "The Migration of Products From Lead User-Innovators to Manufacturers," MIT Sloan School of Management Working Paper (No. 4554-05, 2005)

[22] Summarized in Von Hippel, *The Sources of Innovation*, and Von Hippel, *Democratizing Innovation*.

thus be expected to resist change even more than normal bureaucracy.[23] To be precise, it's not that military processes are resistant to technological change *per se*, but rather that, as Janowitz points out, "the process of innovation in the military establishment itself has become routinized."[24] These formal procurement routines, based on a clear distinction between technology user and provider, tend to select against uncontrolled user innovation (in ways discussed later).

Furthermore, there is a quite different legal and structural relationship between a user and a manufacturer than there is between military personnel and acquisition systems: the former are independent actors in a market while the latter are part of a military hierarchy. Military personnel operate within a highly regulated, high-risk environment that constrains resources available to them. They remain in specific jobs for only a few years at most before transferring. They are evaluated for some war-fighting specialty other than technology development. Finally, "The Professional officer corps' emphasis on hierarchy, the chain of command, obedience, and loyalty, make challenge from below somewhat less probable than one might find in a non-military organization."[25]

Given these considerations, if user innovation theory can successfully predict and explain significant user innovation *even* in an environment as idiosyncratic and seemingly hostile to user innovation as the military, that would provide strong evidence in support of it.

The Functional Sources of Military Innovation

Before proceeding on to that test, it's important to first characterize the functional sources of particularly military innovation in more detail. Whereas Von Hippel distinguishes manufacturers and users, the appropriate functional distinction in the military case would be formal procurement from defense contractors on the one hand and operational uniformed personnel on the other. This is an ideal distinction, for in reality the military employs civilians as well as uniformed personnel, reservists as well as active duty, and there are many types of contracting entities ranging from private corporations,

[23] Barry R. Posen, "Organizations and Innovation," MIT Security Studies Program Working Paper (2001)
[24] Morris Janowitz, *Sociology and the Military Establishment, Revised Edition* (New York, NY: Russell Sage Foundation, 1965), 21.
[25] Posen, "Organizations and Innovation," 23.

non-profit federally funded research and development corporations (FFRDCs) like MITRE and RAND, to government laboratories, workshops, or arsenals. Nevertheless, the idealized functional sources of significant technical innovation can be characterized as (1) the formal separation of use and production managed through a detailed requirements-based contracting process, and (2) the intimate involvement of technology users in the production of technology.

The acquisition system is a sprawling tangle of regulations, congressional relations, military program management, and defense contracting. While costly and inefficient, the system was instrumental to American preeminence during the Cold War, a contest of weapons development with the Soviets, rather than production as in WWII.[26] It provided the U.S. with the large economies of scale needed for sophisticated hardware requiring lots of money, space, labor, and time to field; with technical expertise in advanced technologies and systems integration; and ensured testing, certification, maintenance, and support mechanisms to move from development to operation. The end of the Cold War, however, reduced the threat that justified the system's extent without, unfortunately, much reducing its long-standing costs.[27] These costs include the gross inefficiencies and potentials for failure or scandal that follow from, on the one hand, rent-seeking within the "iron triangle" of congress, defense contractors, and the military,[28] and on the other, the tremendous uncertainties associated with fielding complex weapon systems in an evolving strategic environment.[29]

The classic defense industrial problem, drawing on Mancur Olson's insights about the concentration of powerful interests and under-representation of diffuse interests,[30] is that congressional pork-barrel politics and contractor business interests lead to the acquisition of expensive systems the military doesn't really need. Procurement can unfairly advantage or be captured by contractors. Formal regulations can limit the entry

[26] Harvey M. Sapolsky et al., "Security Lessons From the Cold War," *Foreign Affairs* vol. 78, no. 4 (1999): 77-89.
[27] Eugene Gholz and Harvey M. Sapolsky, "Restructuring the U.S. Defense Industry." *International Security* vol. 24, no. 3 (2000): 5-51
[28] Gordon Adams, *The Politics of Defense Contracting: the Iron Triangle* (New York, NY: Council on Economic Priorities, 1981)
[29] Rosen, *Winning the Next War*
[30] Mancur Olson, *The Logic of Collective Action: Public Goods and the Theory of Groups, Second Edition* (Cambridge, MA: Harvard University Press, 1971)

of competitors by restricting users to the certified "program of record" and prohibiting other solutions—including user innovations—as unapproved, untested, or unsafe. In the process, the bureaucratic identities of military program managers and technical officers (aircraft maintenance officers, computer systems officers, etc.) can lead them to unwittingly champion a specific contractor solution. A "follow-on imperative" can lead Congress to generate projects just to "keep the lines warm" in case of national emergency.[31] Savvy contractors can abet all these processes by becoming more adept at navigating the procurement bureaucracy than servicing end-user needs; that is, it may be more lucrative to prioritize a core competency in regulatory rent-seeking over defense technology production. A more subtle version of Olson's logic is the under-representation of end-users: formal requirements originate within service acquisition staffs distinct from operational units, with often little more than lip service to the needs of the "war fighter," resulting in expensive systems ill suited to actual end user needs.

Furthermore, the complicated acquisition system generates staggering bargaining and coordination costs. Bureaucratic politics (including interservice rivalry, Joint service logrolling, and reform attempts) consumes time and money, potentially leading to threat inflation, capability duplication, and "gold plating."[32] Contractors spend on lobbying and marketing in glossy defense journals and industry conferences, and the government spends on oversight and legal services. Accurate requirements are hard to get because busy users who are deployed around the world will not immediately benefit their current mission by taking time to articulate long-term requirements, and their task knowledge is tacit and situated anyway, thus hard to formalize, while actual input usually comes from staff officers echelons above and at least a tour away from operational reality. Furthermore, funding is slaved to the Congressional budget cycle, and even the most accurate requirements are subject to change because the evolution of technological and strategic environments is unpredictable in detail.

Simply coordinating thousands of engineers and managers spread across programs, subcontractors, and geographical locations makes for interminable meetings,

[31] James R. Kurth, "The Political Economy of Weapons Procurement: the Follow-On Imperative," *The American Economic Review* vol. 62, no. 1 (1972): 304-311

[32] Thomas L. McNaugher, "Weapons Procurement: the Futility of Reform," in *America's Defense*, ed. Michael Mandelbaum (New York, NY: Holmes & Meier Publishers, 1989), 68-112.

management procedures, and perpetual *PowerPoint* presentations. The peculiarities of software further exacerbate these costs. Brooks' 1975 software engineering classic, *The Mythical Man Month*, points out that software, being nearly "pure thought stuff," is irreducibly: *complex* because abstract components allow for a vast number of states and non-linear interaction; *difficult to visualize* because high-dimensional logical abstractions have no one natural representation; *changeable* because incremental improvements are always tractable and the runtime context is highly variable; and *arbitrary* because functionality is intimately tied to evolving domain-specific user interactions. As a result, software projects are unusually susceptible to getting stuck in a "tar pit" of problems that read uncannily like a description of government projects: late, over-budget, over-managed, plagued with communications problems, out of touch with users, difficult to update, difficult to maintain, disappointing, frustrating. His whimsical summary is "Brooks' Law: Adding manpower to a late software project makes it later."[33] Ciborra similarly observes that "Ethnographic research about the implementation and use of information technology suggests that quite often even in large organizations: leadership is missing; and the technology is drifting, as if out of control."[34]

These inefficiencies collectively add up to large transaction costs for defining new requirements and turning them into usable technology. User innovation theory would expect that motivated, technically savvy personnel would rather just design things themselves if possible. Drawing on tacit, situated understandings of their own requirements, clarity of purpose gained through actual mission performance, and their own technical skill, personnel in low-cost niches might realize immediate, small-scale operational improvements by innovating with locally available resources. Especially with information processing tasks, stereotyped organizational routines can be offloaded onto software programs, providing relatively quick returns to user-developers in terms of error reduction, standardization, and time saved for other tasks that computers can't do, such as training, thinking through contingencies, and anticipating the behavior of enemies

[33] Frederick P. Brooks, *The Mythical Man Month: Essays on Software Engineering, 20th Anniversary Edition* (Reading, MA: Addison-Wesley Publishing Co, 1995), 232.

[34] Ciborra, *The Labyrinths of Information*, 21. The Standish Group's 1994 CHAOS report (http://www.standishgroup.com/sample_research/chaos_1994_1.php) estimated that out of a total of $255 billion spent annually on software, $140 billion was wasted, including $80 billion outright from failed projects; 31.1% of projects were canceled, while 52.7% cost 189% their original estimates.

and allies. Immediate gratification, pride in the creation, and the opportunity to assist comrades in similar situations would provide positive feedback to the inventor, encouraging free-revealing and ongoing refinement by the user community.

While these are the familiar expected benefits within user innovation theory, there are unfortunately also some limitations and pitfalls that the theory doesn't dwell on. While present in many organizations, they loom especially large in militaries. Personnel can be legally directed or prohibited to use a particular technology or procedure by their chain of command, which will tend to favor tested and certified solutions over user expedients. For example, all software on military computers is supposed to be tested and certified by configuration managers in an attempt to control security and interoperability externalities that could result from amateur or malicious software. This constrains the size of the sandbox in which users can play, incidentally leading to many user-developed applications being disguised as *Office* documents since more powerful programming packages are not part of the certified configuration.

Coordination problems emerge when users don't have the time, ability, or inclination to standardize and support their inventions. Personnel rotate through jobs frequently, remaining in a given position at most only a couple of years. Combat attrition might, furthermore, eliminate the sole expert. Militaries are always uncomfortable about depending on only one or a few individuals' tacit expertise, so they strive to either institutionalize or avoid idiosyncratic practices. Institutionalization, however, is difficult. Users invent things expecting to gain some immediate benefit in their job, leveraging tacit knowledge of that job gained through active experience. Writing good documentation and training plans—the challenge of making tacit knowledge explicit—is often a secondary consideration that does not provide the immediate fun and payoff that design does. So even though frequent rotation may provide a diffusion channel for user inventions, it also frustrates long-term support for them. Frequent changes of jobs may also make it harder to establish stable user innovation communities.

Military user-developers have "day jobs," an operational task to focus on, and they are evaluated for their competency in this area, not software development. An aviator's career depends on flying not programming. Technology design is not considered a core

war-fighting skill, so superiors and peers are likely to see development and support of technical innovations as a distraction from a member's long-term career interest. This severely curtails the time available for maintenance and support. A final coordination problem is that low-cost components that users incorporate into their solutions may be dependent on external commercial support, which could dry up if a company goes out of business or discontinues a model, leaving the organization in the lurch.

The fact that military information systems handle classified data further advantages bureaucratic control. When an application carries the imprimatur of formal certification, it shifts the risk from local managers to the certifying agency. Managers will resist bearing the risk of security violations or corruption of mission-critical data that might result from informal user innovation. This is likely to result in the over-protection of risk by legitimate managerial authority with respect to the more diffuse benefits of user-experimentation.[35] How much worse for informal improvisation, therefore, when security and interoperability risks are combined?

This long list of problems and potential negative externalities associated with user innovation partially justifies some of the regulation clogging military system management and procurement bureaucracies. The situation is thus that inefficiencies of formal acquisition encourage user innovation, while the risks of user innovation justify formal program management. Table 1 summarizes the relative strengths and weaknesses of each mode of production.

[35] A similar dilemma confronts the intelligence field: counter-intelligence protections (e.g., investigating applicants, controlling data) can be over-provided because the risks of a single mole are quite salient compared to the diffuse benefits of informal intelligence sharing among analysts.

Table 1: Relative strengths and weaknesses of the functional sources of military innovation

	User Development	Formal Procurement
Strengths	Clarity of purpose, enthusiasm Tacit/situated knowledge Low-cost solution Immediate marginal improvement Free revealing, informal diffusion, increasing returns	Economies of scale (time, manpower, money) Technical expertise Lifecycle/configuration management Bureaucratic legitimacy
Weaknesses	Negative externalities (security, reliability, configuration) Coordination problems (documentation, training, scalability, maintenance) Resource constraints (brief tenure, other job responsibilities, technical regulations, undercapitalization)	Under-representation of diffuse user interests Formal requirements out of touch with operational reality (uncertainty, ambiguity, budget-cycle reaction time) Industrial/congressional rent seeking (including marketing, proprietary technology) Steep bargaining & coordination costs (budgeting, contracting, requirements-creep/gold-plating, management, testing, fielding); increasing with "Jointness"

Anticipating Military User Innovation

The primary hypothesis of user innovation theory is that relative expectations of innovation-related gain among different functional actors determines the functional source of innovation. This implies that variations in expectations should translate into variations in the source of innovation. In the military environment, expectations will result from a combination of the above considerations. The challenge in any particular case is to understand how changing supply and demand side factors affect this complex balance.

All things being equal, changes in the properties of technology or user knowledge that lower barriers to entry should promote user development. Rapid technological change or strategic uncertainty should exacerbate the weaknesses of formal procurement, contributing to unmet demand and growing transaction costs for dealing with acquisition systems, encouraging user development. User innovation theory furthermore expects that lead users (who are the first to experience needs in an emerging market, and are poised to

benefit from finding a solution) will be the ones to innovate; that they will freely reveal solutions to other users; and that they will form distributed innovation communities to realize increasing returns.

Conversely, the inability of user-developers to overcome the risks—real or perceived—of informal production should strengthen formal procurement. Unchecked configuration management bureaucracies can be expected to become more adept over time at anticipating and identifying weaknesses in user development and suppressing it. The bureaucratic politics theoretical perspective,[36] with respect to technology acquisition, would expect the procurement system to (1) maintain the essential distinction between user and producer, (2) to reduce uncertainty through formal program management, (3) to increase resources for formal programs, and (4) to enhance autonomy by defining programs within well-understood boundaries.

These contrasting expectations highlight the importance of organizational context for military user innovation. Although user innovation theory describes ways that changes in the supply of or demand for technology might promote user innovation, military bureaucracy can be expected to stiffly resist movement to a new equilibrium. User innovations in the face of organizational resistance could give rise to a wide range of outcomes. Most likely would be suppression or withering on the vine, with inventions abandoned when the inventor transfers. The minimal level of organizational support would be co-optation into existing bureaucratic processes, since less organizational change would be required for procurement managers to simply treat user prototypes as aberrant sources of requirements for formal procurement projects. This may be inadequate, however, if the conditions which favored user development over formal procurement in the first place persist; that is, if cooptation does not reduce unmet demand and transaction costs. More creative (and thus less likely) organizational effort would be required to support ongoing user innovations in that case. Whereas user innovation communities seem to arise somewhat spontaneously in Von Hippel's commercial

[36] Morton H. Halperin, *Bureaucratic Politics and Foreign Policy* (Brookings Institution Press, 1974), James Q. Wilson, *Bureaucracy: What Government Agencies Do and Why They Do It* (New York, NY: Basic Books, 1989), Graham T. Allison and Philip Zelikow, *Essence of Decision: Explaining the Cuban Missile Crisis* (New York, NY: Longman, 1999)

examples, determined military user-developers will have to work hard to find a way to incubate inventions and build organizational support for a user community. Significant bureaucratic maneuvering to support innovation is not included in the original presentation of user innovation theory.

Testing Theory with *FalconView*

The history of aviation mission planning software provides a natural laboratory to examine military user innovation. It spans two decades during which both the technological and strategic environments changed rapidly: the commercial PC-based information technology sector exploded, and the U.S. military became involved in a wide range of operations after the Cold War. Thus barriers to software production fell while warfighting problems changed, providing large variation on user innovation theory's independent variables that influence relative user and manufacturer expectations of gain.

At the same time this case has factors that appear to favor both users and formal acquisition. On the one hand, aviation is a technology-intensive activity that non-aviators never fully understand in detail; aircrew are technically savvy, their education often beginning with an undergraduate engineering degree and continuing with extensive specialized military training; and aircrew are responsible for their own intricate pre-flight planning. Thus, the technological environment, aircrew knowledge, and user accountability for mission planning all promote informal innovation. On the other hand, aviation is a tremendous capital investment for an organization and requires a sophisticated defense industrial base to support. Mishaps are costly, not only because an aircraft and its aircrew represent significant investments, but also because crashes and weapon delivery errors create high collateral damage costs. Large risks make testing, certification, maintenance, safety, standardization, and oversight important, and this empowers formal institutions. Military aviation is thus a domain with potential for user innovation, but also with formidable bureaucratic structure.

During this time traditionally procured systems evolved side by side with informal user efforts. This interaction generated important consequences for the larger military community, including the emergence of a software application called *FalconView* that subsequently became the *de facto* standard for digital mapping across the U.S.

Department of Defense (to include ground forces, special operations, intelligence, as well as manned and unmanned aviation). By providing personnel in all services with an intuitive and inexpensive digital mapping package that improved planning efficiency and promoted interoperability and adaptability, *FalconView* was a technical innovation in its own right, but more importantly it also involved organizational innovation. *FalconView* enthusiasts introduced a different way of developing military software, involving operational users in technical production, and providing tools and support for on-going, decentralized adaptation to novel situations.

During the past decade, operational users drove *FalconView* development and diffusion far beyond its initial scope as an F-16 mission planner, resulting in a versatile platform for networked planning and operations, all for a total cost of about $20 million. By contrast, the formal software acquisition program inspired by and intended to replace *FalconView*, the Joint Mission Planning System (JMPS), remains millions of dollars over-budget, years behind schedule, and had yet to field a certified operational version as of early 2006. Over $1 billion will be spent during JMPS' first decade, to say nothing of the opportunity costs associated with acquisition practices ill-suited to software development. Even then JMPS will remain an aviation-only program lacking important military-wide capabilities that have evolved subsequently in *FalconView*.

This narrative covers two general phases of military user innovation and the acquisition bureaucracy's reaction to it. The first is the emergence of automated mission planning for fighter aircraft in the 1980s, culminating with the user-led development of *FalconView* in the 1990s. The second phase is user innovation activity that builds upon *FalconView* as a foundation as it diffuses to military communities beyond fighter aviation. Through historical process tracing it should be possible to examine whether technical and strategic variation catalyzes user innovation in the ways user innovation theory expects—by altering relative expectations of gain, empowering lead users, and supporting innovation communities—or whether the idiosyncratic bureaucratic context plays a more important role in explaining this case. As mentioned above, a case can be made for the strength of both influences in the military aviation domain.

Sources for this history include interviews and correspondence with key participants, identified through interviewee referrals, official and unofficial primary source documents, either in the public domain or provided to the author by participants, and the author's experience using *FalconView* while an active-duty naval officer. Several interviewees played a key role in both formal and informal programs—an important fact about this case, as discussed further below—and so were in a good position to report reliably on events; this helps mitigate against a potential pro-*FalconView* bias in the interview sample.

Automated Mission Planning

Aviation mission planning includes all of the information gathering, processing, and production aircrew must do before take-off. Simply a cross-country flight requires aviators to: find proper charts; update the status and frequencies of airfields and navigation aids; create a communications plan; compute aircraft-specific preflight and flight performance data; file a flight plan; gather weather data; calculate fuel consumption; and generate knee-board cards summarizing this data and strip-charts (chart segments for the entire route). An attack mission is even more involved, including all of the above plus: parsing the air tasking order from higher headquarters; target area study with imagery; weaponeering calculations; weapons load-out planning; air-refueling planning; aircraft formation and friendly ground force deconfliction; threat intelligence and analysis; avoidance and suppression of enemy air defenses; and combat search and rescue planning. Different missions have different planning requirements, such as slow-down points and computed air-release points for the C-130 airdrop mission. Some missions require more effort than others.

Prior to automated mission planning systems, these calculations were a time-consuming activity involving paper charts, protractors, dividers, whiz wheels, grease pencils, flight manuals, and paper orders and updates. Turn radii were traced around coins, strings pulled across charts to measure distances, and strip charts cut with scissors and glued together. It was not unusual to find some of these practices ongoing in some squadrons as late as the mid-1990s.

In the late 1970s, aviators were experimenting with automating some of these tasks using microcomputers and engineering calculators, just then becoming available in the commercial market. The first mass-produced microcomputers—the Apple II, Tandy Radio Shack TRS-80, and Commodore PET (Personal Electronic Transactor)—appeared in 1977. The IBM PC, running Microsoft DOS, followed in 1981. Hundreds of new firms with no connection to the existing corporate software industry sprang up, and the software industry's growth curve turned sharply upward in what one historian describes as a gold rush. Some of the most successful products *for* microcomputers at that time—the VisiCalc spreadsheet, WordStar word processor, dBase II database—were written *on* microcomputers by groups of just one or two programmers.[37]

Just as these affordable machines opened up a whole new commercial market, they created similar new opportunities for military aviators. "Mission planning is not rocket science," one aviator explained, "but it's complicated with lots of moving parts."[38] Because many of the computations performed manually were well standardized, they were ideal candidates for a computer algorithm. Programming computers to perform oft-repeated and stereotyped calculations essentially off-loads the computational burden onto the machine, reducing errors, saving time, and freeing humans to focus on other "moving parts," especially more intangible considerations about mission scenarios or enemy actions that are ill-suited to an algorithm.[39] In the late 1970s and early 80s, it was both feasible and natural for military aviators, often electronics enthusiasts in their spare time, to obtain microcomputers with their own money and begin writing software to aid mission planning.

A Tale of Two Systems: PC and Unix

Two young A-10 pilots, Captain Jake Thorn and Captain Jerry Fleming, were stationed at Mertle Beach Air Force Base in 1981. The two were avid electronics hobbyists and HAM radio operators who began writing software on a TRS-80 Model 1. The commander of Air Force Tactical Air Command (TAC), General Wilbur L. Creech,

[37] Martin Campbell-Kelly, *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry* (Cambridge, MA: MIT Press, 2004), 202-221.
[38] Jake Thorn, interview by author (13 October 2005).
[39] Edwin Hutchins, *Cognition in the Wild* (Cambridge, MA: MIT Press, 1995); Frank Levy and Richard J. Murnane, *The New Division of Labor* (Princeton University Press and the Russell Sage Foundation, 2004)

met the pair during a routine visit and was favorably impressed, asking, "Can you flight plan on that thing?" Thorn assured him that he could and was invited to give a demonstration in a month's time at a *Corona* meeting of Air Force general officers. The Captains' squadron commanding officer agreed to take them off the flight schedule so they could get to work. The result was "Jake and Jerry's Two Week Flight Planner," which impressed the crowd enough that Creech created the TAC Small Computer Program to provide every TAC squadron with a microcomputer and begin "squadron automation."[40]

TAC purchased Cromemco System 2 (CS-2) machines, providing some minimal infrastructure for every TAC squadron to be able to write and run its own software.[41] Various aviator-written programs percolated throughout the community, passed around on floppy disks from one aviator to another as they commuted around air bases, an effective form of pre-internet software diffusion. One program that became widely used was FPLAN ("Flight Planner"), first written in 1983 by Thorn and Fleming together with another aviator, Captain J. C. Thompson, at Eglin Air Force Base, Florida. FPLAN performed basic route and fuel calculations, but its real strength was an updatable database of navigation coordinates and frequencies from the Instrument Flight Rules (IFR) supplement. Like a garage-start-up software company, the officers' wives manually typed in the data from each update so they could make copies on floppy disks and mail them out to other aviators. Over the next several years, they released several new versions of FPLAN updates on the side while working other primary jobs, and the update distribution list grew considerably. FPLAN found its way onto computers of squadrons across the Air Force and Air National Guard.[42]

---

[40] Thorn, interview (13 October 2005). General Creech later asked Thorn, "If I wanted to see flight planning when I first visited, could I?" Thorn answered, "No sir, but I knew I could do it in a month," displaying a risk-accepting optimism common among user-developers. "I thought so," replied Creech.

[41] Interestingly, IBM did not submit a bid for the TAC contract, and as a result squadrons had CS2s rather than PCs until 1987. They were stuck with the CS-2 long after the commercial market had embraced the PC, foreshadowing the tendency of military IT infrastructure to lag behind the commercial state of the art. TAC also let a contract to Atlantic Analysis Corporation to design some basic mission-planning software for the CS-2, but this withered away from neglect as aircrew used their own programs.

[42] Thorn, interview (13 October 2005)

FPLAN's emergence at Eglin is significant because in 1983, TAC established a center for testing and evaluating automated mission planning systems there.[43] This meant that FPLAN emerged from the one location where contractor-produced mission planning systems also first entered the force. In a pattern that would repeat itself, the same aviators involved with official systems were also developing unofficial systems on the side, having both strong interests in mission planning as well as inside knowledge of the problems plaguing the official effort. These first official solutions, interestingly enough, also used the CS-2 platform, but they ran a Unix operating system while the squadron machines ran a version of DOS. The first Mission Support System (MSS I) included a digitizer table "big enough to double as a bomb shelter," on which a paper chart could be overlaid and coordinates registered with a mouse-like pointing device, enabling some very basic geometric calculations to be performed.[44] MSS I had a lot of problems and was difficult to use as delivered, leading to a complete software overhaul at Eglin in 1983.[45] Captain Thorn was the Operations Officer of the 4485th Test Squadron at Eglin where MSS I was in Operational Test, during the same time he was working on FPLAN.[46]

MSS II followed in 1986, an even larger refrigerator-sized minicomputer system enclosed in its own ruggedized container that needed four people to lift it and a C-130 to transport it.[47] Aircrew preferred to use FPLAN and other informal programs whenever possible to avoid MSS II. To be sure, MSS II had capabilities the informal software did not. These included some of the first digital map displays, which necessitated Unix minicomputers because PC microcomputers were still too underpowered at that time.[48] More importantly, MSS II was needed to load a data transfer cartridge (DTC), a "brick"

[43] Jake Thorn, Mission Planning History Slides (2005).
[44] Thorn, interview (13 October 2005)
[45] Mike Bartgis, "AFMSS MPS and PFPS News," ANG/DOOM Newsletter (May 2000), 6.
[46] Thorn, interview (13 October 2005)
[47] John Pyles, correspondence with author (26 October 2005). The MSS II actually consisted of two computers in the same ruggedized mil-spec container, a MicroVAX for map image processing and a Cromemco S-100 bus, a holdover from MSS I.
[48] John Pyles, interview with author (28 October 2005). These maps were CIA World Databank II vector ("stick") maps at the 1:500k-1:1M scale, as well as some Common Mapping System (CMS) raster ("picture") maps scanned from aviation charts. The British firm Fairchild Defense supplied the original MSS raster maps, prompting the Air Force to develop its own CMS scanning and encoding format to escape foreign dependence. At a 12:1 compression, this format required a lot of storage and graphics power, so PCs were not even an option in the 1980s.

which was loaded by mission planning computers in squadron operations centers and then plugged into a jet on the flight line.

The advent of improved capability aircraft with onboard computers to aid navigation and weapons delivery was a major factor increasing demand for mission planning software. The F-16C, fielded in 1981, had improved avionics and the first data cartridge. With only a 8Kb capacity at first, these could be programmed in the cockpit in a pinch, but by 1990 capacity had grown to 128Kb, and there were some things that could only be done on an MSS II.[49] As more aircraft were fitted with data cartridges, MSS-II was expanded to cover other TAC aircraft, raising coordination costs considerably for program managers.

Unfortunately, not all squadrons with F-16Cs had an MSS II, and those that did found them difficult to move, use and maintain. Air National Guard (ANG) squadrons were particularly hard pressed on both accounts, being well below active duty squadrons in priority for new equipment, and needing to be as portable as possible for deployments. This led ANG Major Walter Sobczyk and Major Robert Sandford to set out to build a device that would hook up directly with a PC and load the cartridge. The first prototype was literally carved out of wood at Hill Air Force Base in Ogden, Utah, and dubbed the "Ogden Data Device" (ODD). The design was refined in-house at the Ogden Air Logistics Center, and soon ODDs and PCs running FPLAN were available to ANG aircrew, eliminating the need for MSS II for the bulk of F-16 planning needs.

ODDs crept into active duty squadrons, provoking the ire of TAC, which perceived them as unsafe and a threat to MSS II, even though aircrew found them to actually be more reliable and hassle-free than MSS II.[50] Major Sandford—who carried business cards with the slogan "Fly fast, fight dirty, and cheat when necessary"—championed the ODD in both ANG and active duty channels.[51] Officers and squadron commanders were willing to ignore TAC protests and use the preferred ODD/FPLAN solution. This reflects a tradition of aircrew responsibility for performing their own planning, but also a degree

[49] Mark A. Gillott, "Breaking the Mission Planning Bottleneck: A New Paradigm," Air University Air Command and Staff College Research Report AU/ACSC/099/1998-04 (1998), 4-6.
[50] Pyles, correspondence (26 October 2005)
[51] Mike Bartgis, interview with author (28 October 2005)

of autonomy for TAC fighter squadrons that might not exist, for instance, in Strategic Air Command (SAC) bomber squadrons equipped with nuclear weapons.

The popularity of PC solutions among aircrew led TAC to direct in 1989 that MSS II mimic the FPLAN interface, and that further programs developed for the Unix MSS II also be concurrently developed for the PC;[52] basically, user innovations were influencing requirements for formal systems. At this time, having just returned from a tour with the Air Force Thunderbirds, now Major Jake Thorn was the Chief of the Tactical Air Force Mission Support System Projects Office at Eglin.[53] Again this individual was a key link in the interaction between the formal Unix and informal PC efforts, managing Air Force spending on the former while still participating in the development and distribution of the latter himself. Although working to improve the formal systems, he was painfully aware of the problems plaguing their development and the importance of ensuring operators had something that worked well in the mean time. These twin paths—official, expensive, something-for-everyone, difficult-to-mange Unix solutions, versus informal, run-on-a-shoestring, mission-focused, PC solutions—would characterize mission-planning software throughout the 1990s.

The 1991 Gulf War provided a major stimulus to automated mission planning. Averaging 2,697 sorties per day for 43 days, the war subjected tactics and equipment to a rigorous "live fire operational test and evaluation."[54] Many aircrew were exposed to digital mapping and imagery on the MSS II for the first time and found that it could reduce both planning time and errors. However, there were never enough to go around, and the machines were frequently down. Many squadrons did not have an MSS II, and the ones that did had only one to go around for two-dozen jets. Many squadrons lacked even PCs. Manual planning with paper charts and grease pencils was still common. Mission planning took at least six to eight hours and sometimes days for complex strikes; target imagery and intelligence was often unavailable; coordination with widely dispersed

[52] Gillott, "Breaking the Mission Planning Bottleneck," 7.
[53] Jake Thorn, military biography page (2005)
[54] Richard J. Blanchfield et al., *Gulf War Air Power Survey, Volume 4* (Washington DC: U.S. Government Printing Office, 1993), 88, 153-170; Lewis D. Hill et al., *Gulf War Air Power Survey, Volume 5* (Washington DC: U.S. Government Printing Office, 1993), 234.

members of strike packages was very difficult; and coordination with friendly ground forces sometimes impossible.[55]

The war highlighted the importance of improving automated mission planning. The Air Force established a System Program Office at the Electronic Systems Center (ESC) to create a common planning system called Air Force Mission Support System (AFMSS). With a much more ambitious scope than TAC's already broad MSS II effort, AFMSS was supposed to provide planning capability for fighter, bomber, airlift, tanker, and special operations aircraft, as well as headquarters command and control. This effort to make "something for everyone" at one go would prove quite frustrating for users and managers alike, as the expense and coordination challenges tended to separate engineers and users with many layers of bureaucracy.[56] The resultant hardware suite, called a Mission Planning System (MPS), would be *even bigger* than the already large MSS II. Exacerbating portability and fielding problems, the entire system was classified, even its generic navigation functionality, since the system also could display intelligence threat data. The classified segment was used infrequently, in part because supporting intelligence organizations and systems were still not well integrated. By contrast, programs like FPLAN had always been unclassified and easy to disseminate.[57]

Major Sandford, working on the National Guard Bureau staff in 1992, was pessimistic that AFMSS would ever be viable for the Air National Guard (ANG). Above and beyond the design problems he anticipated from the cumbersome contracting process, he knew AFMSS MPS would be even less mobile than MSS II, a drawback for deploying Guard units, and furthermore, the Guard likely wouldn't receive them for years after

[55] Thomas A Keaney and Eliot A Cohen, *Gulf War Air Power Survey Summary* (Washington DC: U.S. Government Printing Office, 1993), 177-178; Jane Glaser, "When Reflex is a Matter of Life and Death," expanded draft version of chapter 21 in Bill Gates, *Business @ the Speed of Thought: Succeeding in the Digital Economy* (New York, NY: Warner Books, 1999).

[56] AFMSS development exemplified the "gold plating" (also called "requirements creep") profile described by McNaugher, "Weapons Procurement," 95: "Early optimism about cost, performance, and schedule, combined with inflexibility during development itself, makes it virtually inevitable that the services will field 'gold-plated' weapons—that is, weapons whose capabilities are not cost-effective. Projects are approved on the basis of optimistic cost-effectiveness assessments. Costs turn out to be higher, often substantially higher, than was expected. This calls into question the cost-effectiveness of some of the performance called for in the original requirement. Yet normally little of that performance is sacrificed; indeed, performance requirements may be augmented during development. And by the time information is available to inform real cost-effectiveness judgments, the project has acquired a momentum that makes cancellation difficult."

[57] Gillott, "Breaking the Mission Planning Bottleneck," 8-12.

active duty squadrons. While FPLAN on PCs with ODDs had worked passably as a stopgap, FPLAN was nevertheless reaching the limits of its design, with architectural legacies dating back to the earliest microcomputers (like single character variables) and several ports across different operating systems.

Sandford envisioned a complete PC alterative, running on a single laptop, displaying digital maps, supporting graphical mission planning, and loading cartridges via an ODD. Prevailing on ANG staffers, Sandford obtained unallocated year-end money.[58]

The Killer App

An easy-to-use PC mapping program seems like a natural development: there are lots of imaginable military, government, commercial, and recreational applications for a fast, cheap, PC mapping program; it would seem to have a generic niche like the spreadsheet or word processor. However, nothing arose in the commercial market to fill it like Lotus 1-2-3 and WordStar did theirs. For one thing, PCs were viewed as far too underpowered to support graphics-intensive mapping. For another, digital mapping requires digital map data, and that requires established encoding and compression standards and detailed, regularly updated coverage over a wide area. What data did exist on the market was expensive, and it had to be purchased piecemeal, thus the customers who could afford it could also afford a more powerful Unix machine to work with it. The low-end PC mapping market was thus rather undeveloped in the 1980s and early 1990s.[59] Sandford's application, however, would be perfectly positioned: military users could get access to

---

[58] Pyles, correspondence (26 October 2005); Robert Sandford, interview with author (12 October 2005). The fact that money was available for Sandford's experiment highlights the role of slack resources in innovation.
Pyles' notes from a briefing around this time, reportedly by Sandford, read: "If the federal government gave each AF squadron $500K to spend on mission planning hardware and software, would the squadron commander buy:
A) One (1) AFMSS
B) Two (2) MSS IIs
C) A 486/50 Notebook, ODD, Laser printer, and CD ROM for every pilot and save the $250K left.
It is time the government started making buying decisions like individuals."

[59] ESRI did release a PC version of ARC/INFO in 1986 ("ESRI History," http://www.esri.com/company/about/history.html), but like the company's minicomputer products, it catered to a specialized market of professions like foresters and urban planners, who like the military, usually had better access to government map data than the public did. ESRI's products were typically very powerful, but also very expensive and difficult to learn. The low-end PC mapping market remained undeveloped until proliferation of hand-held GPS devices increased demand in the late 1990s-early 2000s and companies like MapQuest and Google made comprehensive, detailed, updated map data freely available over the internet.

voluminous map data that was being developed for military mapping applications (on Unix platforms), which took care of the latter problem; and PC performance was improving by a factor of two every 18 months, so the former problem would take care of itself.

Sandford contacted engineers at MIT, Stanford, and the Georgia Institute of Technology about the project. He received a lot of resistance to the idea of doing F-16 mapping software on a PC, reflecting an engineering bias for choosing more powerful Unix systems for graphics-intensive computation.[60] Sandford countered that even if PCs weren't fast enough then, they would be soon enough, probably by the time the software was written. He foresaw correctly that Unix minicomputers would remain scarce, expensive assets for squadrons, while PCs running Microsoft software would only become more readily available and accessible. In spring 1993, Sandford found a sympathetic ear at the Georgia Tech Research Institute (GTRI), the non-profit applied research arm of the university, in a research scientist named John Pyles. While a graduate student at GTRI, Pyles had worked on some Army intelligence software that displayed photographs of maps on PCs, which he saw had great advantages of portability and usefulness in the field, as well as on a MSS-II project where he became familiar with military map data formats.[61] Like other engineers, Pyles wanted to do Sandford's project in Unix, even a PC running Unix, but Sandford was adamant that *FalconView* would be a PC DOS-based program. Pyles recruited two more GTRI graduate students onto the team, Robert Gue and Vincent Solicito.[62]

The contract was signed in September for a "Flight Plan Simulation Mapping System," with the word "simulation" added so that the ANG could piggy-back on an existing Army Research Labs contract with GTRI for simulation software. This allowed them to bypass the hassle, delay, and expense of establishing a new contract. GTRI

[60] Sandford, interview (12 October 2005). The engineering resistance was understandable. Not only were Unix machines pervasive in scientific and engineering environments, many attempts by large corporations to create graphics-heavy windows operating systems for the PC had failed. Only with the release of Microsoft *Windows* 3.0 in 1991 had the software matured enough and PCs sped up enough to make graphical windows based programs really feasible (Campbell-Kelly, *From Airline Reservations to Sonic the Hedgehog*, 250-251).
[61] Pyles, correspondence (26 October 2005)
[62] Pyles, interview

managers also required Pyles to phrase the wording of the contract so that deliverables would be "beta"—working prototypes rather than finished product—because they felt the 6-month deadline was too ambitious. "We…were just too naïve to realize this ourselves and just went ahead and finished the initial release anyway," Pyles recalls, "Being fresh out of college, $475K sure sounded like a lot of money to us and as a result we were under a good deal of self-induced pressure to deliver." The beta version was completed in January 1994, and a tested version 1.01 was released to F-16 aircrew in June.[63]

FalconView is essentially a digital version of the classic military map board with transparent overlays (e.g., one for terrain features, another for enemy disposition, another for friendly scheme of maneuver, etc.). Presented in a familiar Microsoft *Office* style with buttons, menus, and a large graphical workspace, *FalconView* could display a wide variety of maps and imagery at different scales;[64] its custom overlays could be saved and shared as small files;[65] it included a connection for hand-held GPS receivers and a real-time moving map display; and also a tactical radio feed to display intelligence data and

[63] John Pyles, correspondence with author (7 December 2005). The fresh-out-of-college naivety expressed itself in other ways, too. Someone on the *FalconView* team included a copy of the video game *Doom* on the CD with version 1.0. *Doom* remained on the CD for three years until a squadron intelligence officer called GTRI and told them he was having trouble getting *FalconView* on his machines because security managers told him *Doom* wasn't accredited (Pyles, interview).

[64] Users can pan around and zoom in and out through different scales of maps and imagery while the overlays move and scale with them. Moving the cursor over the map provides coordinates and elevation accurate enough to slew optical sensors onto a target. Map, image, and elevation data generally is loaded from CD-ROMs provided by the National Geospatial-Intelligence Agency (NGA, formerly known as the National Imagery and Mapping Agency, NIMA) onto local hard-drives or network servers. Greater area coverage, especially at large scales and with imagery, can require a lot of storage, so users generally only load up data they need, although deployable map servers with terabytes of data are now available. Maps were CMS for *FalconView* 1.0, CDARG by 2.0. The smallest scale available at that time was 1:250,000 scale Joint Operational Graphic (JOG) charts; 1:50,000 terrain land maps and 1:25,000 city graphics are now available for selected areas. *FalconView* now also can display USGS and NOAA GeoTIFF and Digital Nautical Chart (DNC) formats, and users can geo-rectify their own custom-scanned imagery. Imagery was initially SPOT. Now unclassified commercial satellite Controlled Image Base (CIB) is available at 10 meter resolution world-wide, many places at 5 meter, and selected areas at 1 meter.

[65] There are different kinds of overlays: routes, threats with range rings, local points for targets and navigation aids, and free-form vector drawings for airspace annotation. Each overlay is saved as a small file, somewhat like an *Office* "document," so it can easily be moved via floppy disk or email. The separation of generic map data from mission-specific overlay files helped not only in data and storage management, but also in keeping *FalconView* itself and the basic map data unclassified. *FalconView* could be run on a classified PC and thus manipulate classified overlay data, while all development could remain unclassified. This gave *FalconView* a tremendous advantage over AFMSS MSS and TAMPS (which could only be operated in a classified environment), making it easier to disseminate without classification controls, and available to user groups with limited security clearance, such as many allies.

friendly force locations.[66] "We knew it would be a really cool app if anyone would let us build it," said Pyles. Furthermore, the constraint of designing for the Intel 80486 processor (rather than a more powerful minicomputer) in 1994 forced GTRI engineers to come up with fast and efficient rendering techniques. Although this constraint evaporated as PC hardware performance improved, it paid dividends later on in fast, smooth performance. "I knew technology was on our side and we could ride the hardware wave of hard disks, screen resolution, and PC performance improvement," Pyles said.[67] AFMSS engineers wedded to Unix solutions did not appreciate that, as Sandford pointed out, "PCs are disposable."

Sandford and Pyles consciously based *FalconView* management on Frederick Brooks' *The Mythical Man-Month*.[68] Brooks argues that conceptual integrity is the most important and elusive part of software design (which, as mentioned above, is characterized by complexity, non-visibility, changeability, and arbitrariness) and this is best achieved with a "surgical team" having fewer, higher quality programmers, a chief programmer controlling the concepts, and frequent end-user interaction to ensure that the task and solution are coordinated. Pyles played the role of chief programmer, while Sandford articulated user requirements and remained closely involved in architecture and interface discussions, sometimes even coding working software to convey a concept. Although a staff officer at the National Guard Bureau, Sandford was still flying regularly (a practice since disallowed for Pentagon staff officers), so his understanding of planning requirements was kept fresh.[69]

Developing an effective human interface is difficult for engineers to accomplish working only from formal requirements and lacking an intuitive understanding of the user situation, a situation common in formal acquisition programs. By contrast, *FalconView* developers made the interface a priority. Two early decisions were particularly important. First, they bet on Microsoft: "it will be whatever Bill Gates decides." This meant designing for DOS and later *Windows*, writing in the C++ language rather than ADA (the

[66] These feeds were not added in the initial F-16 version because the aircraft already had a map display built into the cockpit, but they were some of the first modifications in version 2.0.
[67] Pyles, interview.
[68] Sandford, interview (12 October 2005).
[69] Mike Bartgis, correspondence with author (28 October 2005)

Department of Defense standard at the time), and adopting the interface "look and feel" of Microsoft *Office* applications in order to make it easier for users already familiar with *Office* to learn *FalconView*.[70] They gave a lot of attention to the graphical user interface because they wanted a novice user to be productive with daily tasks right away. In stark contrast to MSS and AFMSS MPS, which required users to memorize cryptic combinations of keyboard commands, *FalconView* became an application that people like to play with the first time they're first exposed to it. As one aviator commented, "I've learned more in 10 minutes on my own with this software than I did during two weeks of training on the MPS."[71]

Second, version 1.0 would be designed only for F-16 mission planning (hence the name "FalconView" after the F-16 "Fighting Falcon"). Even though there was interest in Sandford's project elsewhere, especially the C-130 lift community, he wanted to avoid the kind of requirements creep that plagued MSS-II and AFMSS, which in trying to be all things for all aircraft delivered poor solutions to everyone at great expense. Sandford and Pyles supposed from the start that *FalconView* could be broadly useful for other aircraft, instructor stations in simulators, and non-aviation users, so a challenge would be holding theses other communities at bay.[72] They felt it was essential to first achieve success with the basics: "do only 80%, but do it perfect."[73] This would set a precedent for small incremental changes in *FalconView*, always making sure that basic functionality was solid before trying to add more, and willing to be flexible on deadlines to do so (an informal convenience the large programs didn't share). By focusing on the human interface rather than just technical functionality, and by limiting the initial scope to the F-16, Sandford and Pyles laid the foundation for an application that others—and not only aircrew—would find intuitively attractive.

Growing Pains

FalconView development encountered resistance right from the beginning. While friction between *FalconView* and its programmatic competitors might be expected, there

[70] Joe Webster, interview with author (25 October 2005)
[71] Gillott, "Breaking the Mission Planning Bottleneck," 15.
[72] Webster, interview.
[73] Sandford, interview (12 October 2005)

was also chafing between proponents of different user efforts, reminiscent of the squabbling that can erupt among insurgent factions. In particular, Major Sandford and Major Thorn developed different opinions about the relationship between user-led software efforts and the formal programs. Sandford was exasperated with the inefficiency of MSS and AFMSS, and just as he had pioneered the stand-alone ODD-FPLAN solution, he sought to bypass the programs altogether with an independently funded Air National Guard (ANG) system. Thorn, by contrast, who had been officially involved with MSS-II management as well as informally nursing FPLAN, saw user software efforts as supplementary stopgaps. Thorn hoped to reform the acquisition programs, while the more revolutionary-minded Sandford wanted to avoid them altogether.

The two could at least agree that FPLAN needed to be completely overhauled, Sandford because he needed the flight planning capabilities for *FalconView*'s graphical display, and Thorn because a stopgap was needed until the AFMSS Unix solution was ready. The AFMSS program was completely opposed to attempting to display maps on a PC, thought to be an unpromising and wasteful distraction from the Unix effort. Sandford would thus provide some of his ANG funding for an FPLAN replacement, dubbed Combat Flight Planning Software (CFPS), and Thorn would achieve reluctant AFMSS acquiescence by promising keeping it separate from *FalconView*. CFPS development began at Eglin under a contract to the Tybrin Corporation, where Jerry Fleming, one of the original FPLAN developers, now worked as a civilian.[74] Thorn's DOS-based CFPS would share the same interface with the Unix-based AFMSS systems, so they could co-exist, to the point that AFMSS eventually began funding further CFPS development. This also allowed CFPS to leverage Aircraft Weapon Electronics (AWE) software, also written at Eglin, to facilitate platform-specific cartridge loading.

Sandford's vision for *FalconView*, in contrast with Thorn, was a threat to the AFMSS program. The first version of *FalconView* could interface with CFPS, effectively providing CFPS with the map display that AFMSS would not allow. Faced with the increasingly capable combination of *FalconView*, CFPS, and the AWEs—

[74] Bartgis, "AFMSS MPS and PFPS News," 6-7; Robert Sandford, interview with author (20 December 2005)

collectively dubbed Portable Flight Planning Software (PFPS)—AFMSS and Thorn tried to slow its development. They demanded that route coordinates entered through CFPS remain fixed in *FalconView*; however, Sandford's next version allowed users to graphically construct and modify a route. AFMSS also demanded that *FalconView* not be able to build threats, an expensive classified capability of the AFMSS MPS; the next version allowed users to graphically build threat overlays, a capability that could remain unclassified because the functionality was independent of classified threat data.[75]

The *FalconView* team routinely ignored angry AFMSS protests as they looked into adding new features. They could get away with this for two reasons. First, the AFMSS MPS was encountering a lot of development problems and receiving extremely negative user feedback, so even program managers recognized that a functional stopgap really was needed. Although Thorn wanted to see AFMSS succeed, he began to appreciate the growing utility and potential of *FalconView*. More importantly, however, *FalconView* was shielded from direct AFMSS and active duty Air Force influence by sponsorship by the ANG and Air Force Reserve.

The Reserve, like the Guard, enjoyed a lower AFMSS priority than the active duty Air Force, so funding further improvements to *FalconView* was likewise sensible. This funding was obtained by Lieutenant Colonel Joe Webster, an aviator on the Air Force Reserve staff. Webster took over detailed *FalconView* management in 1995 when Sandford transferred to Nellis Air Force Base, Nevada, to head the F-16 Test Team. Sandford still spoke frequently with and made monthly visits to GTRI, ensuring that *FalconView* development benefited from immediate user feedback.[76] Reservist involvement in *FalconView* development also opened up an avenue for ideas about civilian technology development to enter the military.[77]

By early 1996, *FalconView* version 2.0, a *Windows* 3.1 application with significantly expanded functionality, was percolating throughout the Air Force, aided by Eglin's production of AWEs for more and more different types of aircraft. A tragedy in Croatia further raised its visibility. On 3 April, an Air Force CT-43 (Boeing 737) carrying U.S.

[75] Bartgis, "AFMSS MPS and PFPS News," 7.
[76] Sandford interview (12 October 2005).
[77] Glaser, "When Reflex is a Matter of Life and Death."

Commerce Secretary Ron Brown and 34 others crashed on instrument approach to Dubrovnik, killing all aboard (Kozaryn, 1996).[78] Air Force investigators opined that had the pilots used the *FalconView* moving map in the cockpit, the controlled flight into ground might have been avoided. Later that year, the Air Force made *FalconView* use mandatory on all Distinguished Visitor aircraft.[79]

Frustration with AFMSS mounted and the popularity of *FalconView* grew, spreading informally to an estimated 13,000 users by 1997.[80] The issue came to a head at a meeting of Air Force general officers at the Pentagon. The AFMSS Special Program Office director was fired, and AFMSS was directed to officially incorporate PFPS, although GTRI and Eglin would continue actual development. AFMSS would still put money into MPS because there were some missions and aircraft, such as the F-117 stealth fighter, that were dependent on it, but nonetheless PFPS finally had official endorsement, as well as $4.4 million of AFMSS money over the next two years.[81]

That same year *FalconView* was recognized as one of five finalists in the "core business" category of the *Windows* Word Open, an annual competition sponsored by Microsoft to promote *Windows* software development;[82] Microsoft CEO Bill Gates later included a chapter on *FalconView* in a business book singing the praises of information technology.[83] By October 1997, GTRI released *FalconView* 3.0, the first new version since *FalconView*'s victory over AFMSS, updated for the 32-bit *Windows* 95 operating system and including 100-meter Digital Terrain Elevation Data.[84] With this version, *FalconView* really began to diffuse beyond its humble origin.

User innovation theory expects that technological changes that lower barriers-to-entry for lead users will promote user innovation. Aviators like Jake Thorn, Jerry

---

[78] Linda D. Kozaryn, "Air Force Releases Brown Crash Investigation Report," American Forces Information Services (13 June 1996), http://www.defenselink.mil/news/Jun1996/n06131996_9606132.html.
[79] Chris Bailey, "FalconView: Mission Planning Tools at GTRI," PowerPoint presentation (2005).
[80] Department of Defense, "Air Reserve Software Program Wins Worldwide Recognition," Press Release (3 June 1997), http://www.defenselink.mil/releases/1997/b060397_bt285-97.html.
[81] Jake Thorn, interview with author (15 November 2005). $4.4 million was still only a small portion of the overall $32.6 million AFMSS budget during the same time, FY1999-2000, which covered legacy MPS development and JMPS development, shared with the Navy (Department of Defense RDT&E Budget FY2001-2002, PE Number 0208006F).
[82] Department of Defense, "Air Reserve Software Program Wins Worldwide Recognition."
[83] Gates, *Business @ the Speed of Thought*, ch. 21.
[84] Chris Bailey, interview with author (14 October 2005).

Flemming, and Bobby Sandford fit the lead user profile well: they were interested in planning automation well in advance of a larger user population, which discovered the same needs incrementally later and then en masse during the Gulf War; and as operational aviators, they were positioned to benefit immediately from improving their own planning performance. As computers suddenly became affordable for users' private purchase or for small units' discretionary budgets, as the theory expects, lead users pioneered automated mission planning before formal acquisition programs even existed. Only after users proactively demonstrated and disseminated their programs was it clear that "market" potential existed, and thus formal programs were initiated. In the 1980s, the formal programs were able to innovate improvements beyond the production capacity of users, most notably the first digital maps, which still required Unix minicomputers and a determined effort to produce map data. However, the existence of formal programs did not reduce user innovation activity; in fact, the former actually helped expand demand for the latter by exposing more users to the benefits of automation, especially during the Gulf War, and by imposing the high transaction costs typical of the acquisition requirements process. PC hardware and Microsoft software also continued to improve in capability and decline in cost, while formal programs remained wedded to Unix architectures, further catalyzing lead user innovation.

FalconView, nevertheless, was not a pure user innovation. The code was actually written by full-time GTRI contractors, albeit on a not-for-profit basis. The project maintained an intimate working relationship with users from the start, with Sandford and later Webster setting design priorities and occasionally coding prototypes to convey an idea, and with engineers frequently interacting with operators. This arrangement had the effect of drastically reducing the transaction costs involved in joining user need information and technical solution information. Contractors wrote the code, but operational users were very much a part of the conceptual design process, allowing for frequent feedbacks with working prototypes somewhat like the software design philosophy later popularized as "spiral development." While *FalconView*'s organization was quite different than the formalized and bureaucratic processes that procured MSS and AFMSS, it's important to recognize that it still represented a minimal amount of infrastructure in terms of a contracting vehicle, permanent facilities, non-rotating

technical expertise at GTRI, and progressive insinuation into testing and certification regimes. This put *FalconView* somewhere between exclusively user innovation and traditional defense technology acquisition, which, together with protection from the "big blue" Air Force provided by Air National Guard sponsorship, allowed *FalconView* development to capture some of the benefits of user innovation while avoiding its pitfalls (as summarized in Table 1).

Because users had to work partially within the existing acquisition framework to realize their desired innovation, one consequence was disagreement among lead users as to how far this cooptation should go. This disagreement would shape the acquisition community's response to the embarrassment of AFMSS and the ascendancy of *FalconView*. One possibility, advocated by Sandford and Webster, would be to dispense altogether with the massive inefficiencies arising from trying to engineer planning software within the traditional contracting system, and instead invest in the more decentralized user-integrated *FalconView* approach. Another, advocated by Thorn, would be to start from scratch with a new system and a new contract for a robust mission planning system. The latter would be realized with the initiation of new program of record in 1998, the Joint Mission Planning System (JMPS), intended to replace *FalconView*. Before the JMPS decision and its ensuing problems can be explored in more detail, however, it's first important to explain how *FalconView*'s user group continued to grow and push its development in novel directions, JMPS notwithstanding.

An Expanding View

Expansion of the user base well beyond the F-16 community occurred in three stages: first with other Air Force platforms, particularly ANG C-130 airlift and Air Force Special Operations helicopters and MC-130s; later with Navy and Marine Corps tactical air; and finally with non-aviation and non-military communities. One irony is that a small program originally limited to only F-16 support became popular with a very diverse user group, while large programs designed from the outset to support multiple aircraft communities failed even to satisfy their target users. Lacking a guaranteed budget, *FalconView* had to be responsive to user needs and expanded in a decentralized manner only as long as it met them, whereas by contrast, AFMSS contractors could retain

lucrative cost-plus contracts by responding only to formal requirements, which tended to creep ever upward regardless of performance in the field. Formal AFMSS requirements emerged through a bargaining process within acquisition organizations, involving contractors and staff officers echelons above and at least a tour away from unit-level operations, rather than *FalconView*'s user, "the guy flying the airplane right now."[85]

While this history so far has emphasized the Air Force/ANG fighter perspective, there was also a lot of software developed by aircrew in other communities. The airlift community was especially conducive to the emergence of user-developed software: C-130 squadrons had less money and more aircrew than the fighters; their operation tempo was slower and more predictable, providing time to develop solutions for stable problems; and aircraft had room for aircrew to bring extra computers with them in flight. Since navigators already did their own planning, they "could fly with beta software because if they screwed up it was their wings anyway."[86] One ANG navigator, Major Mike Bartgis, tried to encourage FPLAN developers to incorporate C-130 functionality because existing user-developed programs, Low Level Flight Planner (LLFP) and Computed Air Release Point (CARP), had become unwieldy with each port to a new operating system. Unfortunately, FPLAN ran into similar re-architecting problems, and fighter aircrew were also not enthusiastic about supporting transportation. So during the Gulf War, Bartgis wrote a new program that incorporated functionality of LLFP and CARP with FPLAN and loaded data onto the C-130's Self Contained Navigation System (SCNS). He noted, "The first lesson I learned is all this is a lot easier than contractors make it sound."[87]

In 1992 Bartgis joined the ANG Air Staff and began distributing his program to units. While there he met Sandford and Thorn and realized that it would be easier to piggyback on the emerging *FalconView*/PFPS effort than to start a new one, because even within the Guard, which had a lower status than the active duty Air Force, airlift was lower-status still. Willing to wait for the F-16 exclusive version before channeling what limited funding could be found for C-130 functionality, Bartgis became a key member of the

[85] Webster, interview.
[86] Bartgis, interview.
[87] Bartgis, "AFMSS MPS and PFPS News," 6.

automated mission planning community.[88] Similarly, Air Force Special Operations officers found money to support displaying towers and other electronic chart updates (E-CHUM) to support low flying helicopters and MC-130s.[89]

Thus a group of aviators from different communities began to gravitate around *FalconView* development and lay in more institutionalized support for a growing user community. A dozen or so officers held the first Mission Planning User Conference (MPUC) at the Eglin Officers Club in 1993. This became an annual forum to discuss developments in both PFPS/*FalconView* and the formal programs, attendance growing to over 1100 people in 2000.[90] AFMSS representatives attended the meetings, downplaying the significance of the first PFPS 3.0 distribution in 1998 and dismissing *FalconView* as an amateur toy; by 2001 the same speakers were apologizing to the crowd that they hadn't been able to divest themselves of the big Unix systems fast enough.[91] Bartgis on the ANG staff also began a detailed monthly newsletter called "AFMSS MPS and PFPS News" that went out to a growing mailing list with detailed technical, operational, and programmatic detail on both Unix and PC systems. Both the user conference and newsletter again exemplify the trend in mission planning whereby officers supported both formal and informal programs at the same time.

As user innovation theory would expect, aircrew-developed applications were freely revealed, and a user innovation community was emerging. Free revealing is common among military user innovators; in fact, any code written by servicemembers while in government employ is legally government property, whereas contractor-written code is often proprietary licensed software. *FalconView* developers consciously decided the

[88] This cooperation highlighted the fact that "user" is not a monolithic category, since users from one community can misunderstand the requirements from another just as contractors do. When Eglin, with its close relationship with fighter operators, began trying to develop PFPS navigation tools for C-130s, they resorted to unwieldy MSS II interface conventions. Thus a navigator began a program called *Thaedra* in order to express interface concepts to Eglin, just as Sandford had produced prototypes for *FalconView*, but *Thaedra* actually became a fully functional airdrop planner, essentially a work around of the work around! That is, airlift user innovation was catalyzed by having to wait for fighter user innovation to mature. Notes from the *Thaedra* help-file also exhibit the pride that amateur developers have in their efforts: "Thaedra was written by Captain Scott (Scooter) Stephenson…at home, as an unsanctioned personal project. Thaedra reflected over 1,000 man-hours of coding and was comprised of 12,000 lines of object-oriented Pascal code" (Bartgis, correspondence).

[89] Paul Hastert, interview with author (8 October 2005). This capability was fielded in *FalconView* 3.0 in 1997 but still not properly working in AFMSS as of 2005

[90] Bartgis, "AFMSS MPS and PFPS News," 1,7.

[91] Hastert, interview.

application would be free government-off-the-shelf (GOTS) software so that any government employee could have access.

The emergence of a self-reinforcing user innovation community had already begun with the diffusion of software as aircrew traveled around the world for exercises and operations. Yet, in a military environment where users' activity is regulated and users transfer often, more institutional support was needed. For *FalconView*, this was provided through the cultivation of small groups of engineers at GTRI and Eglin with low turnover and close user interaction, eventual cooperation with AFMSS, and most importantly, the longevity and perseverance of key individuals. Even though officers transferred, they moved through aviation commands and took *FalconView* with them, often taking positions within mission planning acquisition organizations. Furthermore, because ANG personnel policy differs from the active duty Air Force, Major Bartgis was able to remain on the Air Staff for eight years, ensuring some continuity and able to wait out obstacles in the active duty force.[92] His role passed to Major Paul Hastert in 2000, an MC-130 aviator in the Air Force Reserve who had written data cartridge-loading software to work around yet another dysfunctional official Unix system called MiniCAMPS. As of 2005, Hastert still worked in the Air Force Combat Support Office, providing a similar kind of continuity.[93]

The Navy acquisition experience with automated mission planning was frustratingly similar to that of the Air Force with MSS II and AFMSS. The Navy Unix minicomputer program was called Tactical Air Mission Planning System (TAMPS). The system itself was so large it had to be hoisted from the aircraft carrier hanger deck and through the floor into the intelligence center above.[94] Having evolved from a Tomahawk cruise missile planning system, the TAMPS contract had passed through three different contractors, and was completely out of synch with aircrew needs. TAMPS routinely failed operational test and evaluations because of both interface and basic functionality issues. Hank Davison, a former A-7 pilot working as a civilian TAMPS instructor at Naval Air Station Lemoore in the 1990s, confessed, "When just planning, I was having a

[92] Bartgis, correspondence.
[93] Hastert, interview.
[94] Hank Davison, correspondence with author (6 April 2006).

hard time coming up with a use for TAMPS. It took more time than with a pile of charts and scissors, which was faster and more accurate for adept planners."[95] Although TAMPS was the only way to load F/A-18 data cartridges, Navy regulations classified damage to the cartridge as a class C mishap, technically grounding the aircraft, so maintenance officers were reticent to let aircrew take the cartridge since aircrew could still plan without it.[96] The introduction of the AGM-84E Stand-Off Land Attack Missile (SLAM) prior to the Gulf War finally made TAMPS use unavoidable, but even for SLAM planning aircrew preferred to use paper charts and informal planning techniques, only then transferring the results to TAMPS to use it merely as a very expensive data cartridge loader.[97]

To get around TAMPS, naval aviators obtained copies of FPLAN during joint exercises with the Air Force. The Naval Mission Planning Systems (NAVMPS) office was also increasingly frustrated with the system because of the cost ($200,000 per workstation), schedule slips, and intense user dissatisfaction. TAMPS was tied to F/A-18 production, which meant that NAVMPS itself actually had a relatively small independent budget and limited leverage. An F/A-18 aviator working at NAVMPS and long-time FPLAN user, Marine Corps Major John "Festus" Bennett, thus saw cheap PCs and PFPS/*FalconView* —free government-owned software—as a viable alternative.

Bennett worked proactively to get the software out to anyone with a Navy or Marine Corps aviation connection, visiting with aircrew and mailing out hundreds of CDs to "beta users" before it was ever certified. Network managers complained that they would find it on one squadron machine and remove it, only to come back and find it on several more later. Bennett decided he needed fleet "disciples" to overcome the criticism of scientists and engineers at Navy labs who were focused on Unix and hostile to "hobbyshop" PC efforts. To build official support, he would ghost-write messages for units to send to the Chief of Naval Operations staff (OPNAV) with glowing evaluations and endorsements of PFPS, creatively manipulating bureaucratic process to build

[95] Hank Davison, interview with author (18 October 2005).
[96] John Bennett, interview with author (19 Octrober 2005).
[97] Davison, interview.

institutional support for the emerging *FalconView* user community. As grassroots support grew, *FalconView* "was somewhere between a cult and a virus."

The entry point into the Marine Corps was Marine Aviation Weapons and Tactics Squadron One (MAWTS-1) in Yuma, which provided early support for Bennett's efforts with a friendly evaluation. From there it diffused to Marine aviation and ground units, as well as Army helicopter units training with them. *FalconView* proved a real boon to Marine Expeditionary Units (MEUs). Deployed on several ships in an Amphibious Readiness Group, MEU planning often involved flying annotated maps by helicopter around to the different ships. With *FalconView*, Marine planners could just transmit shape files between ships, reducing the number of flights. Marine use led NAVMPS to fund some important features in *FalconView*, such as a program called GEORect that enabled users to take a scanned image of a map or image and tie it to geocoordinates in the *FalconView* environment. This allowed Marines afloat to digitize more detailed ground maps than aviators typically needed, and which NIMA was often unable to provide.[98]

The groundswell of user support led to the approval in 1998 of a Navy PFPS distribution channel via NAVMPS. Although now empowered to send out accredited N-PFPS CDs to Navy and Marine Corps units, it was still a very informally run, minimally funded operation. This actually provided Bennett, later joined by Hank Davison and another naval aviator named Jon Drof, with a lot of flexibility to respond to user requests. This increased user enthusiasm because they felt they had a voice in development, unlike with TAMPS. If Bennett and Davison needed Navy-specific applications, they could just add them to the N-PFPS distribution. If new features were needed, they could leverage the existing Air Force PFPS contract and send money, usually in small $100,000 lumps, rather than go through a difficult source selection process.[99]

This sort of decentralized development process facilitated the growth of *FalconView* functionality. The actual requirements and integration was informally coordinated with GTRI and Eglin, which was successful because engineers either were former aviators or

[98] Bennett, interview.
[99] Bennett, interview; Hank Davison, correspondence with author (3 April 2006).

had developed and maintained a close relationship with aircrew. A standing contract with the Ogden Air Logistics Center at Hill Air Force Base provided the vehicle for time and materials contracts. One result of this was that *FalconView*'s future was often uncertain only six months ahead. Most contributions were less than $200,000 (the outlier being a SOCOM contribution of $2 million in 2003), and if the team didn't field something and get good aircrew feedback the program would be in jeopardy. Major Thorn, Major Sandford, Lt. Col. Webster, and Major Bennett were always "shaking trees" for small amounts of money from AFMSS, the ANG, Air Force Reserve, Navy Mission Planning, or other organizations to keep Pyles' team in business. Every three months Pyles would have to call one of the aviators and warn, "We're about to turn into pumpkins! Do you have more work?"[100] The result was something like competitive pressure for continuous improvement, quality control, and responsiveness to end-user needs, unlike typical defense contracts.

As in the fable about stone soup, different organizations that wanted new features contributed money or prototypes, and these improvements then became available to all users who might leverage them in novel ways. For example, the incorporation of digital terrain elevation data (DTED) and threat masking algorithms to calculate the effect of terrain features on radar coverage was funded by Air Force Special Operations so that aviators could predict which surface to air threats could observe or shoot at them, but this feature could also be used by ground forces to predict what terrain *they* would be able to see from observation points. The Navy funded enhancements like digital nautical charts, image georectification, and a 3D flight simulator ("SkyView"). Army and U.S. Special Operations Command (SOCOM) contributions followed in earnest in 2003, adding ground force symbology, tactical graphics overlays, topographic lines, and illumination planning to estimate shadowing based on sun or moon position.[101] The minimal, decentralized, institutional framework for *FalconView* development allowed different

[100] Pyles, interview.

[101] Chris Bailey, "Department of Defense Usage of FalconView," GTRI White Paper (2005), http://www.falconview.org/docs/FalconView%20Usage%20Throughout%20the%20Department%20Of%20Defense.pdf. Army helicopter pilots were eager to use *FalconView* as an alternative to their own dysfunctional Unix system, the Army Mission Planning System (AMPS).

service components to achieve a degree of interoperability and coordination that ostensibly "Joint" programs aspired to, yet without the centralized oversight.

Second-Order User Innovation

The first wave of user innovation in mission planning that gave rise to *FalconView* culminated in official recognition and support for the application. *FalconView* development requirements were certainly handled more informally than in other software contracts, and *FalconView* contractors nurtured a closer relationship with operational users, yet the design of *FalconView* proper became, nevertheless, a contract-mediated, civilian-managed engineering project. With the stabilization of this development, however, a second wave of user innovation extended the application's functionality, and many complementary applications emerged outside of GTRI and Eglin. *FalconView* essentially provided a toolkit, inadvertently at first and later deliberately, that users could use to experiment with novel ways of automating military information processing tasks.

Extensions might be as simple as combining existing features for some unforeseen purpose, as when bomber crews began using commercial GPS antennas and data loggers not only for real time navigation, but also for more accurate post-mission analysis.[102] Many were programs that generated output files that *FalconView* could read and display over maps and imagery, as with a program called *Sensor* that displayed the actual footprint on the ground of an aircraft's forward-looking sensors.[103] Others were low-level utilities that made *FalconView* more versatile, such as Major Hastert's *PFPS Update*, which automated sharing data among different computers when *FalconView* was originally a stand-alone application.[104] Private contractors or government civilians also designed *FalconView* complements, requested by users but funded by the services, for inclusion on the PFPS distribution. Interesting examples include the bird avoidance

[102] Shawn Fleming, "Using FalconView for Situational Awareness and Post Mission Reconstruction," Presentation at Mission Planning Users Group Moving Map Working Group (1999).

[103] Davison, interview. For example, if the forward-looking infrared (FLIR) lens, like the cockpit display screen, is square, the actual area on the ground sensed by the FLIR is trapezoidal because the sensor looks down at an angle. Aircrew could also use *Sensor* in reverse, deforming target imagery to produce a prediction of what would appear in the cockpit display, aiding visual target acquisition. A very similar program allowed bomber aircrew to dynamically plot the expected pattern on the ground for gravity ("dumb") bombs.

[104] Hastert, interview. *PFPS Update* was, significantly, the first user-developed application to be included on the PFPS CD distribution.

model (BAM), which draws from a database of migratory bird routes to create graphical overlays depicting the probability of bird strikes at a given time of year,[105] and *TaskView*, which parses the Air Tasking Order into an interactive *FalconView* overlay.[106]

Some user applications helped bridged data from "stovepipe" systems that didn't communicate with one another, or worked around them altogether. These were often written as Visual Basic scripts within Microsoft *Office* documents to get around configuration restrictions, with the effect of adding custom functionality to both *FalconView* and *Office*. For example, a database application called *Quiver*, designed by an intelligence officer on the USS Kitty Hawk, managed target and threat intelligence data, outputted target and threat overlays for *FalconView*, and automatically generated *PowerPoint* presentations with target imagery for strike briefs.[107] A similar user-developed application called *Vulture* managed Air Force intelligence and operational planning information for an Air Operations Center, sidestepping some inconveniences of another large Unix system (Theater Battle Management Core System, TBMCS). One especially versatile tool called *Excel2FV* made it possible to import a wide variety of intelligence and other data formats into *FalconView*. As an officer managing the battle in

[105] Bartgis, interview. Bird strikes are a serious problem, and have led to the loss of or damage to many high-performance aircraft.

[106] The ATO is a long and complicated document with aircraft and mission pairings and associated airspace control measures. Parsing it out by hand used to take hours. Pamela Bowers, "CrossTalk Honors the 2002 Top 5 Quality Software Projects Finalists," CrossTalk: The Journal of Defense Software Engineering (July 2003), http://www.stsc.hill.af.mil/crosstalk/2003/07/top5finalists.html .

[107] *Quiver*'s inventor, Lieutenant (Junior Grade) Paul Wilt, demonstrated the program at the Naval Strike and Air Warfare Center in Fallon, Nevada, where he teamed up with this paper's author to improve *Quiver* for use aboard all Carrier Intelligence Centers (CVIC). *Quiver* was popular with intelligence personnel and aircrew, but quite unpopular with SPAWAR, the Navy's information technology procurement and management command, since it competed with an expensive Unix intelligence system (GCCS-I3) and was perceived as beyond the control of configuration managers. Despite thousands of lines of Visual Basic code and operating system calls, the application was still nominally "just an *Access* database," so SPAWAR technically could not prevent *Quiver*'s use; indeed, many contemporary user developed applications masquerade as *Office* documents because "real" executable programs are uncertified. Ultimately SPAWAR decided to incorporate *Quiver* features into a future version of GCCS-I3 and wait for its military supporters to transfer to new assignments. As with *FalconView* and AFMSS/JMPS, the programmatic version spent a lot of money but never managed to reproduce the successful features of the unofficial version. Yet unlike *FalconView*, *Quiver* never found an institutional sanctuary—advocates with staying power willing to work both formal and informal channels—to weather the resistance. A version of *Quiver* called *A3* continued evolving in the Naval Special Warfare community. *A3* played a critical role in the management of intelligence for SEAL operations during the 2003 invasion of Iraq, often going through several new "spiral releases" each week as unexpected information management challenges emerged. Unfortunately, history repeated itself yet again as *A3* moved into programmatic channels, changed names, and withered away.

Afghanistan's Shahikot valley during *Operation Anaconda* commented, *Excel2FV* "literally saved lives countless times and was the ONLY way we could've managed the battle situation."[108]

Recognizing the extent of this activity, GTRI first provided deliberate third party programmer support in 2000 by exposing application program interfaces (APIs) in version 3.1, and in 2002 by providing a more robust software development kit (SDK) in version 3.2. End users now had the ability to add buttons, control the *FalconView* display from other applications, and accept input from the *FalconView* interface. This sort of third party development support was already an important market generating mechanism in the civilian software industry. When software companies create powerful macro languages for their applications, like Microsoft's Visual Basic in *Office*, or expose interfaces, modules, code libraries, or web services that other programmers can incorporate into their own applications, this creates opportunities for third parties, including users, to become experts in customization. This expands the market for the original application by enabling it to adapt to domains wasn't designed for, without the original developer having to pay for the work. This of course introduces configuration management liabilities for third party developers, as changes to APIs, for example, can break their applications, and they are limited to only the functionality the original developer decides to expose. Yet on the whole, third-party toolkits are a good way for

[108] William A. Hastings, correspondence with author (28 October 2005). *Excel2FV* author Lt. Col. Paul Hastert writes (correspondence, 13 October 2005), "I had seen for a long time that there was a need to ingest data from external sources into PFPS, but hadn't done anything about it until 9/11. At that point…I was spending a lot of time at the CAOC [Combined Air Operations Center] in PSAB [Prince Sultan Air Base, Saudi Arabia] and definitely saw a huge need for the program. Unfortunately I also discovered that the chance of getting an 'unauthorized' executable on the CAOC network, let alone the [Top Secret] network was limited….I decided to shift…to Visual Basic for Applications (VBA)…the program [Excel2FV] masquerades as an MS Office file (in my case a Word Document)." Hastert teamed up with Maj. Hastings in the CAOC, who writes (correspondence): "We actually didn't start using it until the battle [Anaconda] had started and had been going maybe about 2 days. There came a point where the data we were using was becoming unmanageable…. It was desperation that forced the issue… At one point in the chaos, I went over and found an airman who I had worked with at a previous assignment and I knew was good with computers. I told him to 'crack the code', figure it out, and write down the steps necessary to import the data from excel (using Excel2FV) into drawing files…. At some point, I found Paul and asked if there could be a few modifications. I really didn't expect any results. My experience with new software and the accompanying engineers wasn't positive. It usually took weeks for them to have a solution. By the end of my shift, Paul had worked out what I needed….I think we made mods almost daily to the program until by the end, almost the entire targets shop was using this program like it was going out of style (and wondering how we had done it before without it)…. Wherever I went after that, I preached the gospel of Excel2FV and trained whomever I could. I'd have to say it was pretty incredible…And we didn't need to wait years for an unusable product after spending millions of dollars."

companies to encourage and leverage lead user innovation to create value (Thomke and Von Hippel, 2002).[109]

Explicit third party programmer support in *FalconView* led to an explosion of new applications created by users, many on a very small scale never heard of beyond the unit in which they originated.[110] It also enabled GTRI to keep the *FalconView* team small, never more than twenty people including administrative support, and to offload support for their diverse user population onto third parties, including users themselves.[111] By designing in adaptability, GTRI recognized that it would often have no idea how *FalconView* was being employed. This is a big contrast to large consolidated programs like MSS, AFMSS, TAMPS, and JMPS, which on the one hand attempted to centrally manage support for a very heterogeneous group of aviators, explicitly defining use cases and functionality, and on the other hand provided no allowance whatsoever to non-aviation user groups because they were outside of their aviation-only charter.

These non-aviation users are a diverse lot, actively adapting *FalconView* for purposes well beyond its initial conception as an aviation planner.[112] National Guard military police in Iraq have used *FalconView* to aid debriefing Iraqi detainees who are unable to read maps but can point out locations on unclassified imagery.[113] Unclassified map and satellite imagery also makes it easier U.S. forces to share planning materials with coalition partners, as well as to send CD-ROM batch updates via regular mail.[114] *FalconView*'s moving map, GPS feed, and tactical radio feed have seen wide use in special operations aircraft and tactical vehicles, including strapped onto the gas tanks of All Terrain Vehicles.[115] Further blurring the boundary between mission planning and execution software, users reconfigured *FalconView* to manage Predator Unmanned Aerial

[109] Stefan Thomke and Eric Von Hippel, "Customers As Innovators: A New Way to Create Value," *Harvard Business Review* vol. 80, no. 4 (2002).
[110] Bailey, "Department of Defense Usage of FalconView."
[111] Chris Bailey, interview with author (14 October 2005).
[112] The actual size of the user population is hard to estimate because the application is unlicensed government software. CDs can be copied or loaded onto multiple machines. GTRI estimated 13,000 total users, aviation and non-aviation, in 2000 and 20,000 in 2004.
[113] Jason Black, interview with author (31 October 2005).
[114] Eric Lipton, "3-D Maps From Commercial Satellites Guide G.I.'s in Iraq's Deadliest Urban Mazes," New York Times (26 November 2004).
[115] Sean Naylor, *Not a Good Day to Die: The Untold Story of Operation Anaconda* (New York, NY: Berkley Books, 2005), 162-3.

Vehicle (UAV) operations in combat: whereas remote pilots used to have limited situational awareness through the "soda straw" of the Predator's camera eye, multiple UAV tracks could now be displayed in real-time over detailed satellite imagery of the operational area, on multiple *FalconView* computers anywhere on the network.[116]

Other unusual uses have included recreating geographic conditions to aid aircraft crash forensics, tracking whale migrations for a Navy environmental study,[117] or a tool funded by the Air Force Surgeon General to analyze directed energy blinding threats.[118] Users in government organizations outside the Department of Defense have obtained copies of *FalconView* and contributed new features through the same military contract vehicle. The U.S. Forest Service uses it to plan airdrops of fire retardants and to track the spread of forest fires. U.S. Customs funded capability for tracking small drug-running aircraft.[119] The National Geospatial Intelligence Agency (NGA) funds functionality and distributes a version of *FalconView* to the intelligence community. *FalconView* has been provided to twenty-five other countries through U.S. foreign military sales.[120] Countries that have purchased U.S. F-16s usually receive PFPS/*FalconView* for mission planning, and it also receives non-aviation use, as in Colombia where it's used for intelligence fusion in counter-insurgency operations. China, moreover, may have illegally obtained a

[116] Paul Hastert, "Spiral Development in Wartime," PowerPoint presentation to NDIA (2005). During the Kosovo war, frustrated Predator operators developed the concept "on the back of a napkin" of embedding sensor telemetry in the closed captioning teletext field of the NTSC video feed. Contractors delivered an expensive, unwieldy solution in 2001 that could display only one aircraft at a time on dedicated "PowerScene" and "BattleScape" machines on the CAOC floor. By July 2002, Lt. Col. Paul Hastert had figured out how to adapt *FalconView*'s GPS feed to accept decoded Predator input, using the existing GPS, map, imagery, and 3-D viewer tools in *FalconView* to provide a quantum leap in situational awareness for Predator operators. By that December, they had a GTRI-developed utility (designed originally for HH-60 helicopter input) to translate the serial Predator feed and broadcast it via TCP/IP to multiple *FalconView* machines in different CAOC areas, each secured at different classification levels with different intelligence feeds (By contrast, the contractor solution was limited to the CAOC floor.). By February 2003 they had a new "SuperSplitter" program to broadcast the location of multiple Predators over the secure SIPRNET internet, and display them all on *FalconView* machines anywhere in the world. This user-led development—rapidly, during wartime—was further enhanced and folded into development of *FalconView* 4.0. It's highly unlikely such rapid progress could be managed through traditional contracts; a willingness to experiment and refine incomplete solutions in a realistic operational environment was critical.

[117] T. J. Becker, "Not for Pilots Only," GT Research Horizons (Spring/Summer 2004).

[118] Bailey, interview.

[119] Becker, "Not for Pilots Only."

[120] Hastert, interview.

copy via hackers who broke into computers at Redstone Arsenal to steal, among other things, *FalconView* 3.2 (Thornburgh, 2005).[121]

Many of the second-order user innovations built on the *FalconView* foundation traced a by-now familiar pattern. Some tech-savvy user with operational experience encountered some pressing operational need and rapidly prototyped a solution with the limited resources that his organizational and technological milieu afforded. Frequent design iterations ("spirals") incorporated feedback from operational, sometimes combat, employment. Solutions were freely revealed to comrades, and user/developer enthusiasm was high. Every step of this process they contended with expensive but dysfunctional legacy systems and limitations on development and diffusion imposed by configuration and security regulations. That is, user innovations built on *FalconView* bore a striking resemblance to the user innovations that gave rise to *FalconView* itself. User innovation with information technology could make real contributions to combat power, yet bureaucratic routines continued to treat this as aberrant activity that could eventually be incorporated into formal requirements-based acquisition.

JMPS Up and Down

Even though *FalconView*/PFPS became the primary aviation mission planning system for the Air Force, Navy, and Special Operational Command from 1998 on, although it catalyzed adaptive second-order user innovation well beyond aviation, and despite the fact that AFMSS was humbled with the firing of its director in 1997 and the Air Force endorsement of PFPS, the acquisition community's immediate response was to launch yet another formal program, the Joint Mission Planning System (JMPS). *FalconView* advocates like Sandford and Webster correctly foresaw that the AFMSS and TAMPS histories of mounting expense and inefficiency were about to be repeated, but they were a tiny minority amid the many parties invested in the traditional acquisition system. Thorn's contrasting preference to reform the system from within via a fresh start and a new contract found many more sympathetic allies.

[121] Nathan Thornburgh, "The Invasion of the Chinese Cyberspies (And the Man Who Tried to Stop Them): An Exclusive Look At How the Hackers Called TITAN RAIN Are Stealing U.S. Secrets," *Time* (5 September 2005).

Ironically, the arguments that *FalconView* proponents employed early on to secure needed bureaucratic support (or at least toleration) for its emergence would later be turned against them. To get Air National Guard funding, Sandford had articulated a vision of Microsoft-based mission planning software running on cheap commercial PCs. Accordingly, throughout the 1990s the debate was framed in narrowly technical terms as a contest between hobby-shop PC efforts and unwieldy Unix programs of record. This resonated with Air Force and Navy aviators who were quite frustrated with their clunky Unix boxes. As long as MSS, AFMSS and TAMPS programs were stuck laboring through their commitments to the Unix installed base and with their great bureaucratic weight, this debate played out well for PFPS/*FalconView*, the only viable PC mission planner available. From the perspective of the AFMSS and NAVMPS program offices, there seemed to be an obvious architectural lesson in this: a single PC-based program might be better than struggling with service-specific Unix "stovepipes." From this point of view, acquisition processes seemed adequate; they had simply been acquiring the wrong kind of product. Switching to a better PC-based product, so it seemed, would make *FalconView* unnecessary.

Framing the problem like this, however, downplayed the fact that the secret to *FalconView*'s success was less that it ran on a PC and much more that end-users were intimately involved in its development. *FalconView* requirements were coordinated informally. Engineers writing code were frequently—often daily—talking to aircrew in operational units. Users were actually creating new complementary applications. Success was built up in small increments of working functionality. Such a process would likely have created decent Unix planning software because the problem it solved—reducing the transaction costs involved in marrying up tacit user needs with software development expertise—was not architecture dependent (Moreover, many early CS/2 user efforts *were* Unix programs). *FalconView* designers understood this well, but the stark "PC vs. Unix" distinction was easier to argue in a bureaucratic context than subtleties about transaction costs in software design. The acquisition program offices discounted *FalconView*'s innovative process, however, and focused only on the product: a common PC-based mission planning system.

Even more, program management offices (PMOs) were actively critical of the process. They had built up animosity with PFPS/*FalconView* and were determined to replace it. "The services would use us as a whip against the PMO," John Pyles recalls, because *FalconView* oft provided new functionality on the cheap that the programs said was impossible.[122] The programs in turn argued that *FalconView* was maverick software unsupported by adequate testing, review, and configuration management regimes; it was not scalable, reliable, or interoperable (ignoring that *FalconView*'s wide demand-driven diffusion suggested the contrary). In 1998, right on the heels of the PFPS victory over AFMSS, the Electronic Systems Center (ESC) commissioned an independent Carnegie Mellon study of PFPS. The study recommended that, although it was well-designed software, PFPS should not be the system of record because it was too personality-dependent and therefore unsustainable. The program offices appealed to the Carnegie Mellon report and to austere Defense Department configuration management directives to undermine the renegade PFPS.[123]

On top of this, NAVMPS did not want to depend on PFPS because it was, since its 1997 coup, officially an AFMSS program; essentially, the Navy did not want to replace TAMPS with an Air Force program.[124] Bureaucratic deadlock was dealt with in the usual way, by creating a new over-arching Joint entity. The services were finally digesting the implications of the 1986 Goldwater-Nichols Department of Defense Reorganization Act, and the Joint Chiefs of Staff 1996 "Joint Vision 2010" called for complete operational, technical, doctrinal, and cultural "Jointness." What better way to demonstrate compliance with the new spirit of the times than the creation of a new program?

With the inauguration of JMPS in late 1998, *FalconView* once again officially became merely a temporary stopgap awaiting the development of the larger program of record. *FalconView*'s purported unsustainability risked becoming a self-fulfilling prophecy. Joe Webster and John Pyles had conducted an internal architecture review and were well aware that *FalconView* needed a major revision. Although it ran fast and

[122] Pyles, interview.
[123] Thorn, interview (13 October 2005). Defense Information Infrastructure, Common Operating Environment/Joint Technical Architecture (DII COE/JTA) standards are set by the Defense Information System Agency (DISA) to enforce interoperability and information security among the services; attempting to satisfy excruciatingly detailed DII COE can distract contractors from focusing on user functionality.
[124] Thorn, interview (15 November 2005).

reliably, there were legacies dating back to its DOS origins constraining expansion, and new opportunities in the latest Microsoft NT networked operating system they wanted to exploit. However, all new development money was going to JMPS, and PFPS was officially on life support.[125] Modest lumps of money would continue to flow to PFPS from various parties for specific features, as discussed previously, but none of this supported the kind of deep re-architecting Webster and Pyles sought. JMPS proponents could furthermore point to the *FalconView*'s need for an overhaul as yet another justification for starting anew with JMPS.

Initial hopes for JMPS were high, even among many aviators who had been active proponents of PFPS against AFMSS and TAMPS.[126] Colonel Thorn even returned to active duty full-time to take charge of the JMPS project, hoping to use the fresh start to reconstruct the advantages of *FalconView* with the resources of scale and bureaucratic legitimacy a managed program could provide. The initial goal appeared achievable: simply replicate the existing PFPS/*FalconView* capability on a scalable and extensible PC *Windows* architecture within eighteen months, reusing existing software where practical. Less remarked upon was the fact that JMPS—a Joint program destined for Air Force, Navy, Marine Corps, Army, and Special Operations aviation—had a far more ambitious scope than AFMSS, which likewise had been far more ambitious than MSS II. One unlearned lesson was that coordination costs could be expected to scale steeply upward. A single program for everyone also made requirement creep inevitable, even though it was hoped that scoping initial production to PFPS functionality would control it. It turned out that PFPS proponents' optimism for JMPS would be short lived as bureaucratic habits in the acquisition system proved stubbornly resistant to change.

The JMPS systems integration contract was awarded to Northrop Gumman in 1999. Cursed with a generous $50 million budget to re-implement PFPS 3.1, hundreds of

[125] Bartgis, "AFMSS MPS and PFPS News;" Webster, interview. Joe Webster, correspondence with author (29 May 2006), estimates the needed work would have cost a few million dollars and a new major release of PFPS. Thorn and other JMPS supporters were confident that JMPS would be able to provide the necessary functionality on time, and so resisted even this hedge investment,

[126] "This project holds great promise to break with the problems of the past" (Gillott, "Breaking the Mission Planning Bottleneck," 24); "Since JMPS is based on PFPS, there is no excuse for not meeting the user needs….the user should be completely happy since the user will be on familiar ground" (Bartgis, "AFMSS MPS and PFPS News," 8).

engineers and managers were brought in, interminable meetings were held, an avalanche of slideshows, messages, flowcharts, and production plans was generated, and the project started suffocating with red tape. By 2001, cost and schedule overruns began to threaten the program's existence, so Northrop Grumman started cutting corners to keep it afloat.[127] With multiple subcontractors involved and incessant development problems, the bulk of effort was expended on coordination within the program and the contractors, rather than between users and developers. Things that were easy in *FalconView* became hard in JMPS.[128] Exasperated with the situation, the original three *FalconView* developers left GTRI in 2000, and Thorn retired from JMPS in 2002. A certified, operational first version of JMPS had still not been fielded as of early 2006, six years after the program had originally planned to field a replacement for the 1999 version of *FalconView*/PFPS.

Exacerbating JMPS inability to adequately meet initial requirements, users' operational needs continued to evolve. In Iraq throughout the 1990s, the US conducted No Fly Zone patrols and periodic strikes. In Kosovo in 1999, the US fought the largest air war since Desert Storm. In these operations the bulk of mission planning was digital, and there was a greater emphasis than ever before on time-critical precision strike, which entailed a greater information processing burden. Inevitably, aircrew and staff officers discovered new items for their wish lists. Major combat operations in Afghanistan commencing in 2002 and Iraq in 2003 stressed mission planners in new ways yet, especially given the importance of special operations and ground force coordination in these operations. The *FalconView* user group expanded dramatically beyond the aviation mission-planning jurisdiction of JMPS, as well as to foreign coalition partners who would not have access to JMPS.

---

[127] Thorn, interview (15 November 2005).

[128] A good example of the difference between working with users, leveraging their situated knowledge, and working from formal requirements can be seen in the way the two applications treat data sharing. In *FalconView* this is accomplished through overlay files that are opened like *Office* documents, which is intuitive for users and facilitates easy sharing over email and easy development of complementary applications that manipulate overlay files. In JMPS this is accomplished through an SQL database server and a complex sequence of mouse clicks, which is superior only from a narrow engineering perspective. "You asked for transportability, we gave it to you. The requirement has been met," explains John Bennett (interview).

This flux in the strategic environment was only mentioned indirectly in previous pages, but was as important on the demand side for user innovation as *FalconView*'s characteristics were on the supply side. *FalconView*'s decentralized mode of development—freely available government software with an open architecture friendly to third-party development—promoted adaptation to new problems. By contrast, JMPS response to emerging requirements was unavoidably constrained by a formal requirements vetting system, a complex and closely controlled architecture, and closed contractor-licensed software. JMPS, a Joint service *aviation* mission planning system, also lacked both the means and mandate to respond to non-aviation user requirements.[129] While JMPS was still struggling to field its first version, four new operationally certified versions of *FalconView* fielded as of this writing. Much of *FalconView*'s funding came from other organizations like Special Operations Command (SOCOM) not tied to JMPS. JMPS, like MSS II, AFMSS, and TAMPS before it, had much trouble adequately responding to its original requirements, and even if it managed to, the requirements had usually changed in the mean time.

The future of JMPS and *FalconView* is far from settled. The services appear locked in to JMPS because advanced aircraft like the F-22A, F-35, and F/A-18E/F require it to fly, and because of growing sunk costs and the opportunity costs of not developing alternatives. While improvements have been made to PFPS/*FalconView*, largely funded by mission planning communities in the Army and SOCOM who view JMPS as extraordinarily high-risk, the PFPS architecture is not ready to provide all that is expected of JMPS.[130] One interesting development in 2003 was that the original JMPS mapping engine developed by Boeing Autometric ran into so many problems that GTRI was given a contract to adapt the *FalconView* mapping engine for inclusion in JMPS (Bailey, 2005d). This helps GTRI address *FalconView*'s architectural challenges somewhat and means, ironically, that JMPS will include pieces of *FalconView*, and may even be subsidizing its own competition. The organizational problems with JMPS development, however, appear to be intractable. All the versions of *FalconView* have collectively cost

[129] Paul Hastert, correspondence with author (31 October 2005), observes, "If it doesn't cut a cartridge then they've got no interest, and in many cases thinly veiled hostility that people are using 'their' software for something else."
[130] *E.g.*, real-time weather and web-services (Bailey, interview).

about $20 million since 1993, while JMPS will likely cost well over $1 billion in its first decade. Colonel Thorn, in retrospect, estimated that it should have been possible to deliver everything JMPS was supposed to do for $200 million using the *FalconView* model.[131]

It's an open question whether *FalconView*'s non-aviation supporters—or even Navy and Air Force aviation commands alarmed with JMPS problems—will begin to seriously invest in a new *FalconView*, whether *FalconView* will just persist in its traditional role as perennial stopgap, or whether some new mission planning solution will emerge to displace both JMPS and *FalconView*. Given the prominence of user innovation in mission planning software history, this last possibility shouldn't be dismissed lightly, especially given that advances in cheap, powerful commercial geospatial software (as exemplified in Google's *Earth*) continue to lower technical barriers to user innovation. At the same time, mission-planning software, like command, control, and intelligence software generally, has become a multi-billion dollar defense industry, with all the congressional, corporate, and bureaucratic politics that entails. This will tend to raise organizational barriers to user innovation, since the field is far more crowded and regulated than ever before.

Discussion: War upon the Map

The 19th-century strategist Henri de Jomini wrote, "Strategy is the art of making war upon the map, and comprehends the whole theater of operations. Grand Tactics is the art of posting troops upon the battlefield according to the accidents of the ground, or bringing them into action, and the art of fighting upon the ground in contradistinction to planning upon a map."[132] Strategy flows from the top down in this classic conception, from the map to the territory. Theories of military innovation, likewise, depict civilian leaders and senior officers leading innovative change. However, digital mapping in the military developed in the opposite direction, bottom up from everyday operational concerns, discovering along the way a strategically important capability: lead users provided with powerful toolkits could extend functionality into new domains and

[131] Thorn, interview (13 October 2005).
[132] Baron Henri de Jomini, *The Art of War, trans. by G.H. Mendell and W.P. Craighill*, Project Gutenberg Ebook (2004), 69.

improve operational adaptability. This blurred the distinction between designer and user, and for aircraft with moving maps, unmanned vehicles, and operations heavily dependent on reach-back intelligence, blurred the distinction between "fighting upon the ground" and "planning upon a map."

The history of automated mission planning is largely a clash between lead users in low-cost innovation niches and organizational resistance to change. It is more than a single case, which supports some more confident generalization. While *FalconView* is certainly the most significant application to emerge, other user-developed applications preceded it, developed with it, and were later built upon it. Many user-developed applications emerged as workarounds to problems with an alphabet soup of formal programs (MSS I and II, AFMSS MPS, TAMPS, AMPS, MiniCAMPS, JMPS, GCCS, TBMCS, etc.), or explored totally new areas of functionality that the large programs ignored or said was too difficult. An entire range of user-developed maintenance, scheduling, and other administrative applications has not even been discussed, nor have applications completely outside the domain of aviation.[133] As a class, such inventions are genuine prototypes rather than just local expedients, and as such, they create the possibility for interaction or interference with formal acquisition.

There are indeed idiosyncratic aspects to *FalconView*'s story: the maturing of PC hardware and *Windows* software; the Unix bias in formal programs; the development of critical technical complements like GPS, worldwide map and image data on CD-ROM, and data-hungry weapons systems; aircrew responsibility for their own planning techniques; the lower priority of the Air National Guard and Reserve with respect to the active duty Air Force and AFMSS; the demands and capabilities of the single-seat F-16

[133] Norman Friedman, *Seapower and Space: From the Dawn of the Missile Age to Net-Centric Warfare* (Anapolis, MD: Naval Institute Press, 2000), 217-222, 354n, discusses the example of JOTS, a naval tactical computer decision aid cobbled together for $25,000 by the enthusiastic staff of the USS *Eisenhower* carrier battle group, commanded by Rear Admiral Jerry O. Tuttle, during a large exercise in 1981. JOTS (originally the "Jerry O. Tuttle System," later renamed the "Joint Operational Tactical System") facilitated air and subsurface interceptions, providing more or less the same capabilities as a costly and complex system called Task Force Command Center (TFCC), which had been under development for the past decade by the Naval Electronic Systems Command. Unlike TFCC, JOTS was designed by the staff that would use it in combat, some of the actual software written by sailors, and it soon replaced TFCC throughout the fleet. By the early 1990s, Tuttle commanded the Naval Space and Electronic Warfare Command (SPAWAR, formerly the Naval Electronic Systems Command); JOTS' transition from informal experiment to program of record was complete (Friedman, 2000: 217-222, 354n).

for both fighter and attack missions; key personalities involved in both formal and informal management; the immaturity of military computer network regulations; the end of the Cold War and the destabilizing of operational expectations.

Nonetheless, the persistent repetition of a basic pattern is striking. Formal programs had an extraordinarily difficult time producing quality software, while at the same time the barriers to entry came down for smart users to design software. Traditional acquisition processes remained stubbornly in place, and user development activity, while often effective, was generally viewed as unsustainable and illegitimate. This forced lead users to work within and around the system to develop their applications and to build in some modicum of organizational support for operation and maintenance. This general pattern was traced not only by *FalconView*, but also by simpler applications that preceded it and others later built to complement it.

In the case of *FalconView*, a passionate core of individuals remained actively engaged for several years, laying in an organizational and technical foundation to support diffusion to and adaptation by an expanding user group. The profusion of user-developed functionality it catalyzed becomes most interesting when it's seen not just as a collection of specific information processing applications, but as an improvement of military adaptability in general, a novel feature at the organizational level. That is, the important *FalconView* macro-innovation was not a specific piece of PC mapping software, but the establishment of a versatile toolkit and a user innovation community supporting the decentralized emergence of novel information processing capability across the force. *FalconView* was a user innovation that made user innovation easier, a bottom-up process with macro-level consequences (that have not yet completely played out).

How well does lead user innovation theory explain this case? Given the challenge of meeting user needs (demand) with technological solutions (supply), the theory predicts that changes in supply and demand side factors that shift relative expectations of innovation-related gain among users and acquisition programs will shift the functional source of innovation. If shifted to users, low-cost innovation will be concentrated with lead users who will freely-reveal their inventions and form self-reinforcing user innovation communities.

The two decades studied involved high supply side variation with the emergence of low-cost personal computing in the 1980s, increasing power and flexibility (to include sophisticated third-party development tools) throughout the 1990s, and growing computer literacy in the military population. At the same time there was high growth in information-processing demand with the introduction of advanced capability aircraft and growing planning complexity in Joint operations and post-Cold War conflicts. As expected in theory, falling barriers to production for users who stood to immediately benefit from solutions to emerging information processing problems allowed them to develop the first automated mission planning prototypes. These lead users freely revealed their programs to others. Only later as the market for mission planning became obvious did formal programs begin providing solutions, and then their advantages of scale were able to provide novel functionality, such as early digital mapping and precision-guided weapon support, and standardize training and support.

Furthermore, even after formal programs were established, user innovation continued. Supply-side evolution of information technology continued lowering barriers to user production, aided by the emergence of *FalconView* itself, while user demand continued to evolve faster than the formal requirements cycle could keep up. Exacerbating rapidly changing demand, the formal programs floundered on the difficulties of enterprise software development, further raising transaction costs for users dealing with programs. Thus lead users continued inventing their own solutions and sharing their software, aided by the emergence of semi-institutionalized user communities gravitating around *FalconView*.

High values on the independent variables lead to the expected result on the dependent variable, the functional source of innovation, with the expected characteristics of lead user innovation, so by and large user innovation theory passes this military test. However, detailed process tracing of this case revealed some important conditions and constraints on user innovation in the military. Most importantly, users had little choice but to cooperate to some degree with existing contracting procedures to provide innovations a stable incubation. While small-scale programs like FPLAN might be totally user-produced, GTRI's contract and engineering expertise was essential for supporting a more robust invention like *FalconView*. These user-driven efforts,

furthermore, were often managed in conjunction with formal programs by officers with AFMSS and NAVMPS, as necessary supplements to the beleaguered programs. This partial cooptation provided *FalconView* enough legitimacy to survive certification regimes, and staying power to accumulate a growing user community.

This shows that the distinction between user innovation and formal acquisition can be an over-simplification, because what is important is the organizational context that allows one formal arrangement or another to thrive. *FalconView* could go toe-to-toe with much larger Air Force programs because it was a contract sponsored and sheltered by the bureaucratically separate Air National Guard, and later adopted by key personalities in Air Force, Navy, and Special Operations acquisition channels. *FalconView*'s legitimation, along with certification of Microsoft's *Office* and other programs on military networks, provided an umbrella for second-order user innovation as long as it used these approved components. This also restricted some lead users from pursuing the ideal solutions they might like, as network management regulations imposed sometimes-insurmountable barriers to production, even if technical capabilities existed in the marketplace. Finally, even the *FalconView* user innovation community, while self-reinforcing as the theory expected, required some institutional architecting to support it. Lead users could be quite creative in coming up with solutions for their needs, but they had to remain attuned to organizational constraints and resources as much as technical ones.

Lead user theory generally performs well in a military environment, but it is vital to factor in regulatory constraints and political support and opposition when analyzing expected innovation-related gains. Within an organization, moreover, although some informal user collaboration is possible, a robust user innovation community requires the consent and support of formal organizational authority. Given the organizational support needed for protecting user innovation communities, it's reasonable to expect that increasingly ambitious levels of user innovation would indeed require promotion and protection from senior officers as Stephen Rosen or civilians as Barry Posen would expect. Further research should explore these dynamics more in depth.

In the substantive area of military software development, it should be expected that as military operations grow increasingly complex and computer-dependent, (1) software acquisition efforts managed by traditional procurement processes will continue to fail to provide quality software, and (2) the informal user-developed applications arising in the lacunae will usually be rather immature or stunted in the absence of exceptional platforms like *FalconView* that can catalyze their development. The prevalence of information technology innovation by military personnel itself constitutes an important organizational innovation, making modern information-intensive operations possible, but it is also an incomplete one because it is by and large still organizationally illegitimate.

Solutions to this challenge will require re-conceiving enterprise software acquisition to more actively incorporate lead user innovation rather than merely collecting formal user requirements. It's beyond the scope of this paper to offer any detailed theoretical or policy analysis.[134] Nevertheless it's likely that the intensive growth of information processing activity and technology in the military will necessitate new conceptions of military production, personnel, and organization. As Dallas Irvine pointed out with respect to a previous generation of military information technology:

"The art of war therefore became, in a sense that it had not been before, an art to be pursued upon the map, and with an immensely greater number of permutations and combinations possible than ever before. Obviously, the conduct of war upon this level required a far different order of intelligence, knowledge, preparation, and skill than the command of a visible mass of men upon a visible terrain."[135]

[134] Gompert, et al., *Extending the User's Reach*, is one recent attempt to analyze the pathologies of Department of Defense information technology acquisition and the promise of user innovation, but the authors' major recommendation is, inexplicably, to further centralize requirements with U.S. Joint Forces Command. Hope springs eternal that ever more "Jointness" might treat the acquisition blues, but this is surely a cure worse than the disease, being antithetical to the decentralized nature of user innovation.

[135] Dallas D. Irvine, "The Origin of Capital Staffs," *The Journal of Modern History* vol. 10, no. 2 (1938): 161-179.