JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 11, No. 10, October 2014

Dynamics of Software Sustainment

Sarah Sheard,* Robert Ferguson,* Michael Phillips,† and Andrew Moore‡
Carnegie Mellon University, Pittsburgh, Pennsylvania 15213-2612

DOI: 10.2514/1.I010169

Trends in sustainment cost growth are beginning to alarm military planners. Although software drives most military functionality, the contribution of software sustainment to sustainment costs is not well understood. The Carnegie Mellon Software Engineering Institute is involved in a research effort to describe the dynamics of sustainment, focusing on the software aspects. This paper describes the development of a dynamic economic model of sustainment in order to predict the consequences of funding decisions within sustainment organizations. To create this model, a number of notions had to be defined, including sustainment capability, capacity, and performance. The initial systems dynamics model uses notional input data; calibration with specific organizations will increase the fidelity of the model and tailor it to those specific groups.

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169

I. Introduction

AIRCRAFT and weapons systems depend on millions of lines of code for essential functions. As these systems evolve over and beyond their initial expected lifetimes, software sustainment has become a complicated issue of significant concern to the U.S. Department of Defense. Studies have shown that the cost of software sustainment climbs as high as 70% of the total cost for the life of the software [1]. A U.S. Air Force study reported, "The resources needed to sustain its legacy aircraft may increase to the point where they could consume the resources needed to modernize the Air Force . . . " and "a budget 'train wreck' with respect to sustainment costs is looming" ([2] pp. 6 and 22). The Deputy Assistant Secretary of the U.S. Air Force established a committee to provide recommendations that would result in long-term reductions in sustainment costs ([2] pp. 3–6).

To date, though, little progress has been made toward understanding how sustainment concerns should be handled when making budget decisions. Although software does not appear in the top 10 budget factors in a 2012 briefing by the U.S. Air Force Materiel Command Deputy Director for Logistics [3], software sustainment consumes 2.8 million hours of effort per year [3]. Why is this number so big?

A more forward-looking question would be the following: How do we know we are making the best possible decisions when we expend funds on software sustainment? This is a complex issue because funding decisions require an evaluation by four different stakeholders: the operational command, the operational concept (CONOPS) and planning group, the life-cycle command, and the program office. It is also difficult for stakeholders to assess the effects of their decisions on processes far removed from their authority and from the time of the decision. In addition, some of these stakeholders have no direct concern with software sustainment, although eventually they are all affected by the performance of the software sustainment organization.

To optimize investments in sustainment, it is necessary to develop a shared understanding of how sustainment factors interrelate. Studies in Germany have shown that decision-making in complex situations can be improved by providing decision makers with simulations that show long-range effects and the downstream fallout of uninformed quick reactions [4]. An ultimate goal of a sustainment simulation is to be able to predict and control sustainment costs earlier in the life cycle, such as during design of the system.

To meet these challenges, it is important to study software sustainment. As platform service lifetimes lengthen, software sustainment is the primary means of extending their operational capabilities. The B-52's current 90 year planned lifetime includes functionality that could never have been imagined by its designers in the late 1940s; this is made possible mostly by software capability expansion.

In contrast with hardware sustainment, which typically focuses on returning an item to its original condition (for example, by replacing broken parts or removing corrosion), software sustainment is intended to change the item. Whether software is changed to correct flaws, improve performance, or adapt to a new operating system, the resulting software is intentionally different from the original software. Because software is highly flexible, the distinction between maintenance and enhancement is often muddied, as is the type of funding that should be applied. If enhancements are large, contracts, requirements specifications, and competitive bidding are required. But small improvements are often called "software maintenance" and are not funded in the same way. This key distinction makes understanding and properly funding software sustainment an enormous challenge.

Simple and traditional economic models, such as return on investment and net present value, are insufficient for software sustainment, where many factors are changing at once and emergent effects can result in sudden and dramatic changes in outcome. The Carnegie Mellon Software Engineering Institute (SEI) is developing and calibrating a system dynamics model for investment in sustainment of software-intensive systems. The objective is to characterize the dynamics of military sustainment, including the mission of the sustainment organization, the demand for sustainment, and the capability of the organization to perform sustainment. The model, which will show the changes in these dynamics that result from funding increases or decreases, will allow sustainment organizations to identify cost implications of decisions and to make adjustments before problems or cost overruns become insurmountable.

The SEI's systems dynamics model describes a number of variables that influence the sustainment process and models the dynamic interactions among them. The final output will be a set of calibrated models that program managers can use to test the effects of changing allocation of funds between product development work and updating the infrastructure of the development organization. The model must be dynamic (that is, it must show time dependency) because demands for software change are frequent and the underlying technology of software products changes rapidly. The model is expected to show that failing to invest early in staff training, tools, and processes will cost so much in time and resources later that the

Received 15 August 2013; revision received 22 November 2013; accepted for publication 23 January 2014; published online 30 June 2014. Copyright © 2014 by Carnegie Mellon University. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. Copies of this paper may be made for personal or internal use, on condition that the copier pay the $10.00 per-copy fee to the Copyright Clearance Center, Inc., 222 Rosewood Drive, Danvers, MA 01923; include the code 2327-3097/14 and $10.00 in correspondence with the CCC.

*Technical Staff Senior Member, Software Engineering Institute, Software Engineering Measurement and Analysis, 4500 Fifth Avenue.

†Technical Staff Senior Member, Software Engineering Institute, Client Technical Services, 4500 Fifth Avenue.

‡Technical Staff Senior Member, Software Engineering Institute, Cyber Enterprise and Workforce Management, 4500 Fifth Avenue.

program may not recover. The models will allow program managers and managers of sustainment organizations to test the viability of several decision scenarios and share the consequences of each decision with senior staff during funding discussions.

II. How Sustainment Works

In practice, individuals are often familiar with either hardware sustainment or software sustainment. Theoretical treatments differ for the two activities and do not integrate them. The organizations performing sustainment must develop their own processes that accomplish both activities. However, to improve sustainment or any other practice, it is necessary to begin with a baseline: a documented description of current practice. This work is a step toward that goal.

Figure 1 shows the interaction of hardware and software in sustainment. When replacing software, there are a number of considerations in determining what to save and what to rewrite. Because the software will have experienced many changes, the original architectural choices are probably long obsolete and no longer satisfy contemporary strategies for design, including modularity, performance, information assurance, coupling, and testability. On the other hand, mathematical algorithms, sensor interfaces, and complicated decision trees should probably be saved. Determining how much redesign the software needs and how much the organization can afford requires extensive cost-value analysis. The problem is no easier for hardware. At what point does it become impractical to modify an engine to achieve greater power? What is the cost-value decision to replace the engine with a more contemporary version when we consider not only the engine cost-efficiency but also the effect of other changes to aircraft design, supply chain, and maintenance procedures? For computer hardware, when do the negatives of low memory and speed become bad enough to risk changing what works for better performance?

In both hardware and software maintenance, common usage distinguishes categories of maintenance activity [5]:

1) The first category is Corrective. Either a defect exists or a part has worn out. In software, corrective maintenance usually involves fixing bugs.

2) The second category is Perfective. A new feature is added or the product is changed to improve performance. A software example is adding new algorithms to improve resolution from currently available sensor data.

3) The third category is Adaptive. The product must adapt to changes in other systems or its environment. Examples include updating aircraft software to provide data required by new ground systems. Adaptive changes are probably the largest single driver in software costs today.

4) The fourth category is Preventive. Hardware preventive maintenance usually refers to replacing parts before they break catastrophically. For software, this is typically information assurance and malware prevention, although it could also mean rearchitecting to improve reliability.

A. Sustainment Value and Sustainment Performance

The value of sustainment is frequently misunderstood. While its cost is easy to calculate, its value is usually described qualitatively or not addressed at all. Modeling sustainment economics provides the opportunity for sustainment organizations to identify not only the cost but also the value of decisions. As part of this complex but capable system-dynamics model, the concepts of sustainment demand, sustainment capability, and sustainment capacity need to be described.

Sustainment value includes 1) readying equipment for operational theater, 2) preventing obsolescence of older equipment by refreshing its technology to enable the equipment to continue to meet warfighter needs, and 3) making costly new-start programs unnecessary, or at least postponing them.

The value of sustaining the infrastructure required for sustainment work includes 1) reduced errors in performing maintenance (both hardware installation errors and software design, coding, or integration errors), 2) higher-quality software (modern tools help enforce coding standards and detect errors early), 3) reduced sustainment costs (providing tools, training, and processes to employees improves work throughput and reduces defects, thereby reducing cost), 4) faster sustainment turnaround time (reduced waiting for sustainers to be available), 5) possibly increased innovation (improvements in tools and techniques), and 6) more capable and stable workforce (keeping employee skills fresh, and lower turnover).

Measuring sustainment performance ideally involves measuring the extent to which sustainment provides value. The proxy used here for sustainment performance is the fraction of existing systems that are considered fully functional out of all systems (which is equivalent to 1 minus the fraction of systems considered to be in need of upgrade). This is not perfect for several reasons, but it does allow visualization of the work queues as they build up and as they get worked off. This measure does not include a notion of cost, so a variable of sustainment productivity is included, which relates performance and cost.

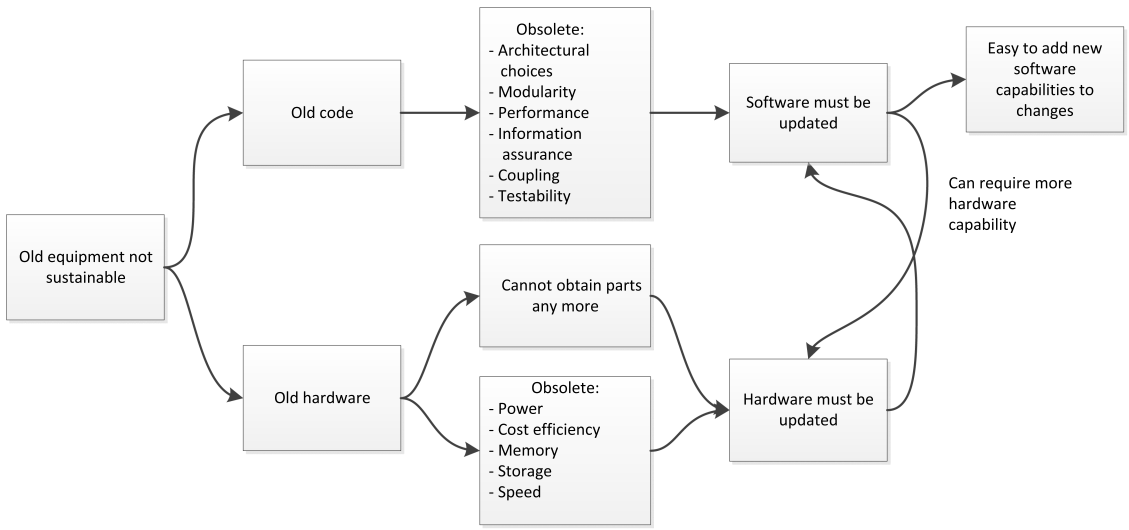

Fig. 1 Interaction of hardware and software in sustainment.

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169

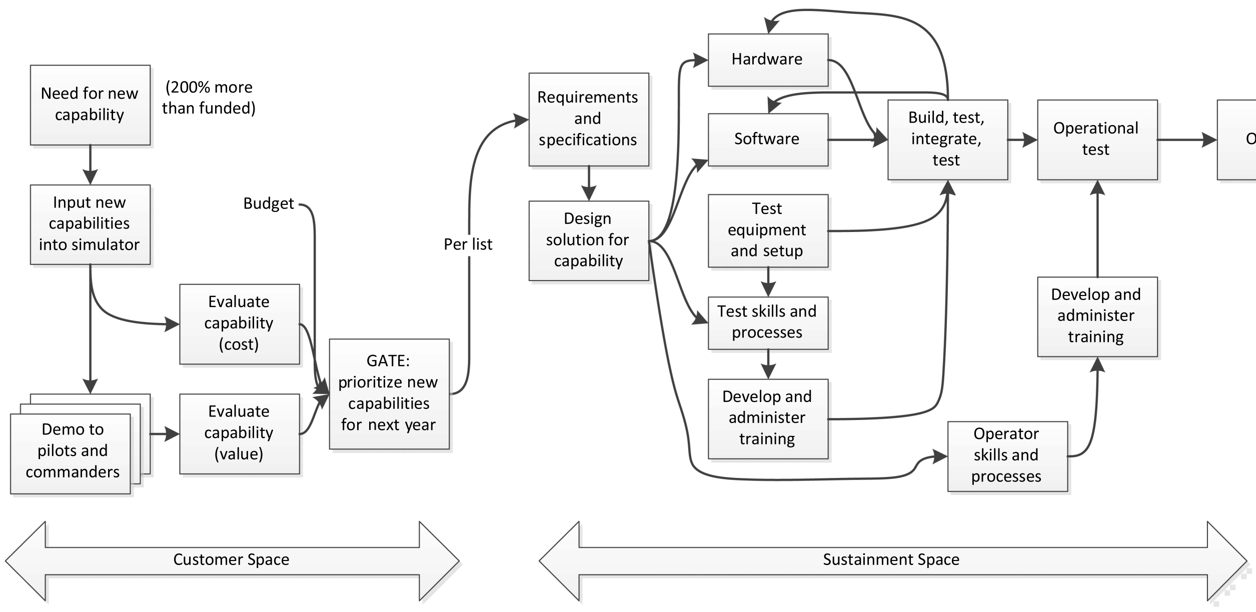
Need for new capability
(200% more than funded)
Input new capabilities into simulator
Budget
Evaluate capability (cost)
Demo to pilots and commanders
Evaluate capability (value)
GATE: prioritize new capabilities for next year
Per list
Requirements and specifications
Design solution for capability
Hardware
Software
Test equipment and setup
Test skills and processes
Develop and administer training
Build, test, integrate, test
Operational test
Operate
Develop and administer training
Operator skills and processes
Customer Space
Sustainment Space

Fig. 2 Sustainment demand to sustainment capacity.

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169

B. Sustainment Demand

Value to stakeholders creates demand for sustainment, and the sustainment organization's stakeholders evaluate how well the organization's performance meets their needs. This evaluation can be expressed in terms of gaps between expectations or wants and the actual performance. Gaps interact dynamically. A gap between desired capabilities and current system capabilities leads to requests for sustainment. A gap between desired and actual sustainment capacity can slow down implementation of improved system capabilities, which causes a sustainment performance gap.

The role of enhancements as part of sustainment is less well understood than maintenance. However, consider how demand for software enhancements comes about. Figure 2 is a schematic of one organization's process for identifying and accomplishing software enhancements. The figure shows both what goes on in the customer or demand space and what goes on in the engineering or solution space (which can also be described as the sustainment space). On the left, demands arrive as requests for new capability. The organization uses a mission simulator along with operational users to determine the effect of different new capabilities on operational missions. This allows them to evaluate the value of each capability compared to the cost to implement it. The organization passes the gate of establishing its next year's budget when it has created a prioritized list of capabilities.

The right half of the figure addresses the sustainment space. Incoming needs are not requirements or specifications; engineers must create those from the incoming desired capabilities. Then, they architect the solution and design and implement the pieces of the solution. These pieces might include hardware and software for the product; test equipment, software, skills, and processes; and operational skills and processes. Training is created to ensure both the integration and test effort and the operational missions can successfully use the new capabilities.

C. Sustainment Capability and Capacity

Software maintenance always involves development tasks, whether those are replacing a few lines of buggy code with a patch, or larger changes such as adapting to a new data description standard. In addition, sustainment organizations must become and stay proficient in integrating new hardware (sensors, armaments, etc.) onto existing platforms; devising requirements, test, and integration processes; training the sustainment staff in such new processes; and providing the software staff with current development tools. Decisions about whether to allocate resources to get a particular update out the door faster or to improve the capability and capacity of the sustaining organization have considerable downstream consequences. Making these consequences clear can potentially prevent premature mothballing of a fleet of valuable mission systems.

In this work, capability is considered to be the skills that individuals have relative to the work that needs to be done. If Java programmers are needed, then some, but not all, software sustainers need to know Java. If a new language is brought in, then instantly the workforce is less skilled than they need to be until a sufficient number learn (and become proficient in) the new language.

Capacity is the count of sustainers multiplied by the fraction who have the required capabilities. This is a simplification of the reality that an organization needs different numbers of different skills, but it correctly models the fact that capabilities must continually be improved. An organization needs both an adequate capability and an adequate count to have full capacity, as shown in the upper left of Fig. 3.

If there are too few employees to do the needed work (lower left), then there is a "count" problem underlying the capacity problem. This can happen if it is hard to hire: if government pay is lower than the pay of other employers in the area, if other employers offer more attractive jobs (like designing computer games) or have what employees perceive as better working conditions (like a shorter commute due to not having to go "on base"), or if government funding is perceived as unstable (in budget crunches and under conditions of sequestration and furloughs). Increased attrition can also cause a reduced headcount. Two ways organizations address count problems are hiring (which is slow and can be subject to hiring freezes) and hiring contractor employees, which can also be slow and can run afoul of the law requiring 50% organic staff [6].

If there are enough employees but their skills are inadequate (either because they have not been in the job long enough to understand the domain or because their skills have become outdated compared to new technology coming in), then there is a "capability" problem, which likewise reduces capacity. Organizations address capability problems with training (or other skill enhancements like job rotation) and, if count is also a problem, hiring skilled employees (if available). Again, delays ensue because training takes time, because becoming proficient takes even more time, and because new hires do not understand the organization and its processes immediately.

D. What Is the Sustainment Infrastructure?

The sustainment infrastructure is everything that makes sustainment happen, aside from the people. This is commonly assumed to mean facilities, cranes, and jigs, as well as a spare parts depot, but those are hardware sustainment concepts.

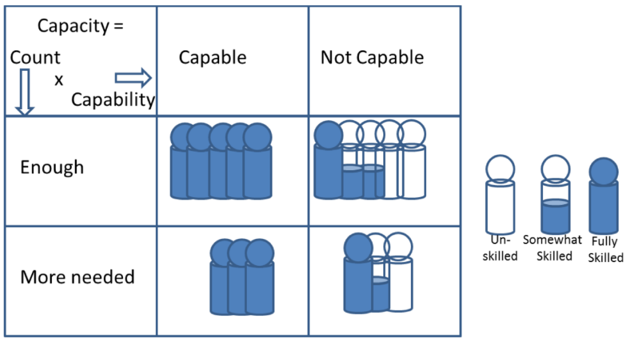

Fig. 3 Capability, count, and capacity.

Software sustainment infrastructure is less visible. Software sustainers also need tools to reduce errors and speed their work. Life-cycle management tools can help track tasks, control versions, and plan increments, as well as integrate with other developmental aids such as collaboration tools, tools for static and dynamic code analysis, and tools for requirements management. Test equipment should include enough actual flight equipment that new software can be tested with flight equipment in the loop and equipment for testing does not need to be scavenged from operational aircraft.

An organization's processes are also considered part of its infrastructure, and they must be maintained. The order in which steps are done can be very important, as well as who signs off on task completion. When new technologies come in, whether tool technology, new languages, or even new flight radars (e.g., a software-controlled radar), the organization's processes must be updated with the proper steps and decision points.

Training is part of the sustainment infrastructure. Keeping sustainers current on languages and development methods is important, but software sustainers must also continually add understanding of new equipment to the knowledge they have built up about legacy software.

The sustainment infrastructure is not cheap, but its function is to enable sustainment and perform it in a cost-effective manner. Infrastructure is generally funded by fees or contributions from ongoing programs and returns value in the form of ready equipment, processes, training, and tools that do not have to be created fresh for every program.

III. Methodology: The Systems Dynamics Model

To build a sustainment model, the SEI team first described the various variables that may affect sustainment performance, such as capability of the staff, number of sustainment requests in the queue, stakeholders, funding cycle, and the sustainment infrastructure. The influences that each variable exerts on other variables are described using a causal-loop ([7] pp. 68–92) formalism, with the variables shown as nodes and their interrelationships shown as arrows. Stocks (variables that can grow and decrease in amount) are shown as boxes, and flows that empty and fill these stocks are shown as pipes with valves [8]. Implications of the causal loops are analyzed and other variables are added that influence the speed of filling and emptying of stocks. These auxiliary variables also influence other variables. Measurement or reporting variables (which depend on but do not influence other variables) are also created.

Once the causal flows are understood, a simulation model is created. For each variable, an equation is created that shows quantitatively how the value of that variable depends on the values of the variables that influence it. The simulation is run, and plots can be made regarding what happens if different input variables change value. Notional variable values are entered and the values of output variables are studied; input values, dependency equations, and relationships among variables are adjusted until the simulation has the expected behavior. Simulations are run that duplicate scenarios in the real world to see if the model works properly in these cases.

When the simulation is running correctly with notional variable values, the next step is to calibrate the model by asking sustainment organizations to provide input about the applicability of the model and then collect actual values of the variables from organizational data. The model and its simulation outputs can then be run with any number of different inputs, either in the form of "what if" analysis or else by running a specific scenario to see what will happen to other variables. Scenarios that are meaningful to users of the model could include new threats, a layoff, a delay in training, or a delay in funding.

The SEI is currently working with one collaborating organization to obtain calibration data. Other potential collaborators in the U.S. Army, Navy, and Air Force have been identified.

The way to use a simulation model is first to represent the normal behavior of a system and then to introduce a new input to see how the responses change. The model developed by the SEI simulates the behavior of the different aspects of the sustainment process, including stakeholder perception, the capabilities of the sustainment staff, and the capacity of the sustainment organization to deliver the work. The system response is being examined to various change scenarios, including the following:

1) Threat: An external change (such as a new threat to the warfighter) results in a request to update the system capability. This request means the sustaining organization will have to update the product and then update processes, skills, and tools to match. It may be necessary to reequip the facility and retrain the workforce (in other words, to sustain the sustainment infrastructure); this requires significant funding. The SEI's systems dynamics model helps decision makers analyze the effect if funding for this improvement is delayed.

2) Support Technology:The sustainment organization decides to improve its own throughput and adopts new processes to "do more with less." Typically, the change is also in response to new quality goals. In this case, the model helps codify the effect on sustainment capability and capacity, and therefore on operational performance.

Figure 4 shows the influence of new technology, which triggers both of the preceding scenarios. New technology can mean new technology in a product, but it also can mean new tools for sustainment.

3) Workforce Considerations: Sequestration effectively decreases the staff available to sustainment organizations by 10 to 20%. How does this decrease affect a sustaining organization's ability to meet its sustainment demand? Does it affect aspects of the warfighter mission as well?

IV. Process Cycles and Loops

Our current model of sustainment includes five basic processes, each of which has inputs, outputs, throughput, and cycle times. The specific inputs and outputs suggest what operational data should be sought to calibrate the model in order to study forces and feedback functions. The processes are listed in Table 1 along with example inputs, outputs, throughput quantities, and cycle times. Processes are a useful way to think about

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169

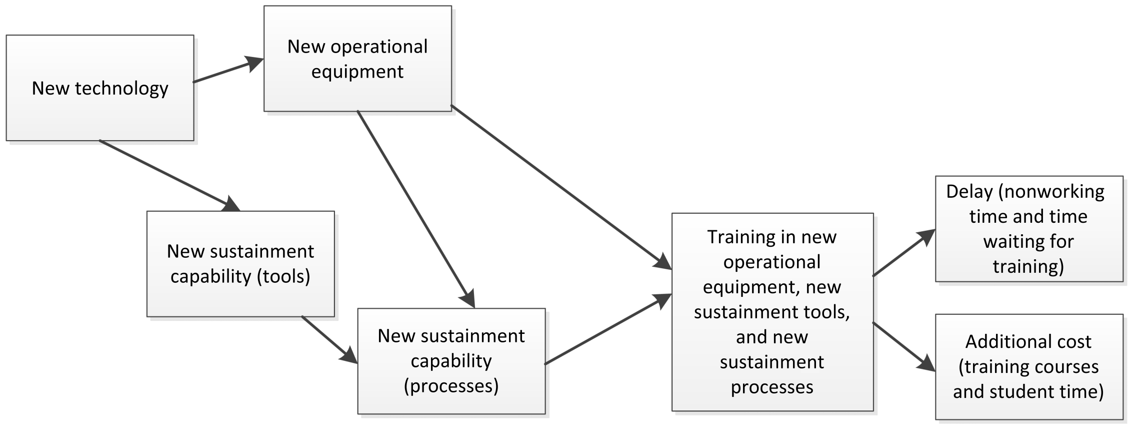
New technology
New operational equipment
New sustainment capability (tools)
New sustainment capability (processes)
Training in new operational equipment, new sustainment tools, and new sustainment processes
Delay (nonworking time and time waiting for training)
Additional cost (training courses and student time)

Fig. 4 Implications of new technology.

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169

the diagram because they divide the complexity of the dynamic model into understandable chunks and their inputs and outputs help structure the gathering of operational data.

Dynamic loops are a traditional way of examining a system dynamics diagram. They show how the influences of one variable propagate through the model and come back to influence the first variable again. Archetypes of types of loops have been described that have predictable behaviors ([7] pp. 378–390).

The next section describes in detail the following dynamic loops:

1) The “bandwagon effect” occurs when successful missions and high mission performance lead to additional demands for capacity and capability.

2) Sustainment work decreases the queue of waiting sustainment requests.

3) The “Limits to growth” loop occurs when the capacity and capability of a sustainment organization limit the rate of completion of sustainment work. As this extends the time required to redeploy, it can result in reduced demand over the long term, or even an operational switch to using an alternate platform [7]. (This is related to the archetype of the same name that includes a reinforcing loop, the bandwagon effect, attached to a balancing loop, which is this one.)

4) “Work bigger” occurs when a sustainment organization may attempt to meet sustainment demand by requiring overtime work or employing extra contract employees. Either of these approaches may work for a short time or a small additional cost, but they stress the organization and quickly reach the limits of their effectiveness. The organization can hire staff, but it must also allow time for training and acculturation of new hires to meet performance objectives.

5) “Work smarter” occurs when a sustainment organization invests in new capabilities (skills, tools, and processes) and possibly additional resources (people and facilities) to improve capacity for sustaining work.

The latter two cycles borrowed heavily from Repenning and Sterman [9], who adroitly showed the short-term loss/long-term gain effect of investing in training.

V. Sustainment System Dynamics Diagram

This section explains the complex system dynamics diagram shown in Fig. 5. The diagram was constructed to study the interactions among sustainment needs and sustainment capability and capacity.

A. Reading Systems Dynamics Diagrams

In Figs. 5–15, quantities of interest to sustainment are shown in boxes if they can be thought of as stocks of items that increase or decrease in number. Flows (double lines) are like pipes that fill or drain stocks, depending on the direction of the arrow. These are shown with valves (hourglass shapes) that control the flow, and thus the rate at which the stock is filled or drained.

In example 1, in Fig. 6, changing technology is a valve that, when open, increases the stock titled “Technology Sophistication.” The cloud means the source of the flow is not modeled.

Some variables are shown outside of the boxes. These auxiliary variables may be inputs, constants, or measurements. They are included because they affect the value of stocks and flows.

Table 1 Process cycles in system dynamics model

Process cycle	Input	Output	Throughput	Cycle time
Operational performance	Missions measured by capabilities used and mission-capable availability	Action reports measured by percent success, and availability gap	Missions performed	Days to months
Operational needs analysis	Mission performance measures and new potential threats, technologies, uses, and mission capabilities	New capability definition	Prioritized operational needs	Weeks to months
Engineering and delivery	Sustainment demand (accepted and not-accepted requests) Sustainment capability required (skills, tools, facilities)	Delivered products by count of deployments and costs; Sustainment gap (requests not accepted)	Sustainment capacity	Hours to months
Capacity and capability development	Changes to training, tooling, facility, processes; Hiring, furloughs, and attrition	Capacity available (percent of request); Capability available date or delay	Capability changes, capacity improvement	Months to years
Improvement funding	Funding requested for capability and capacity development	Time required to fund, amount funded	Funding requests satisfied	Multiple years

Arrows show that the value of one variable is part of the equation that determines the value of another variable. An “S” on the arrow indicates that, when the first variable goes up, the second variable goes up in the same direction (positively correlated); “O” indicates they vary in opposite directions (negatively correlated).

In example 2, in Fig. 7, the rate of “adding expected capabilities” goes up (S) when technology changes faster (arrow from left to right). The rate of adding expected capabilities also goes up (S) when the number of committed stakeholders goes up (vertical arrow).

Loops appear when several variables influence each other: one variable’s changes propagate to other variables, for which the changes then influence the original variable. If the reactions to an initial rise in the first variable cause that variable to continue to rise further, the loop is called a reinforcing loop. If the effect of the loop is to instead cap the rise in the initial variable, the loop is called a balancing loop.

To determine whether a loop is reinforcing or balancing, count the number of times “opposite” (O) effects occur around the loop. If the number is zero or even, then the loop reinforces an initial change. If the number is odd, then the loop balances out the initial change.

B. Reinforcing Loop 1: Bandwagon Effect

The first loop to be discussed is reinforcing loop 1 (R1; bandwagon effect), on the lower left of Fig. 5 and expanded in Fig. 8. On the left, technology is continually advancing, adding to the sophistication available to products, processes, and adversaries. Technology advances increase expectations for system capabilities. This results in more potential stakeholders thinking that this system would be useful, and they turn into committed stakeholders. These stakeholders then identify further capabilities they would like the system to have. The combination of two positive correlations results in a reinforcing loop. (Note that this reinforcement works both ways: if for some reason a number of committed stakeholders withdraw their support, this reduces the expectations for the system’s capabilities and encourages more stakeholders to pull out.)

C. Sustainment Work Loop

Figure 9 shows the system sustainment work loop. On the right side, fully functional systems are shown as a stock. When an upgrade is requested due to a gap between expected and actual (average) system capabilities, then the system is no longer considered fully functional, but instead, it is considered a system in need of upgrade. As sustainment work is done, and those systems are upgraded; they become fully functional systems once again.

Upgrading of systems results in an increased number of implemented capabilities (topmost arrow), and therefore a higher average system capability. This decreases the gap between expected and actual system capabilities. Thus, this loop is a balancing loop.

D. Sustainment Capability, Capacity, and Performance

The sustainment capacity is the power of the organization to perform upgrades: to turn systems in need of upgrade into fully functional systems. The sustainment capacity is defined as the product of the number of available staff and the capability of that staff.

Sustainment capability is defined as what the average sustainment staffer knows and can do; this could include knowledge of languages, processes, platform design and hardware; and even tool familiarity. (The concept of capability can also be extended to include purchase of tools or improvement of processes, but these are not explicit in this model.)

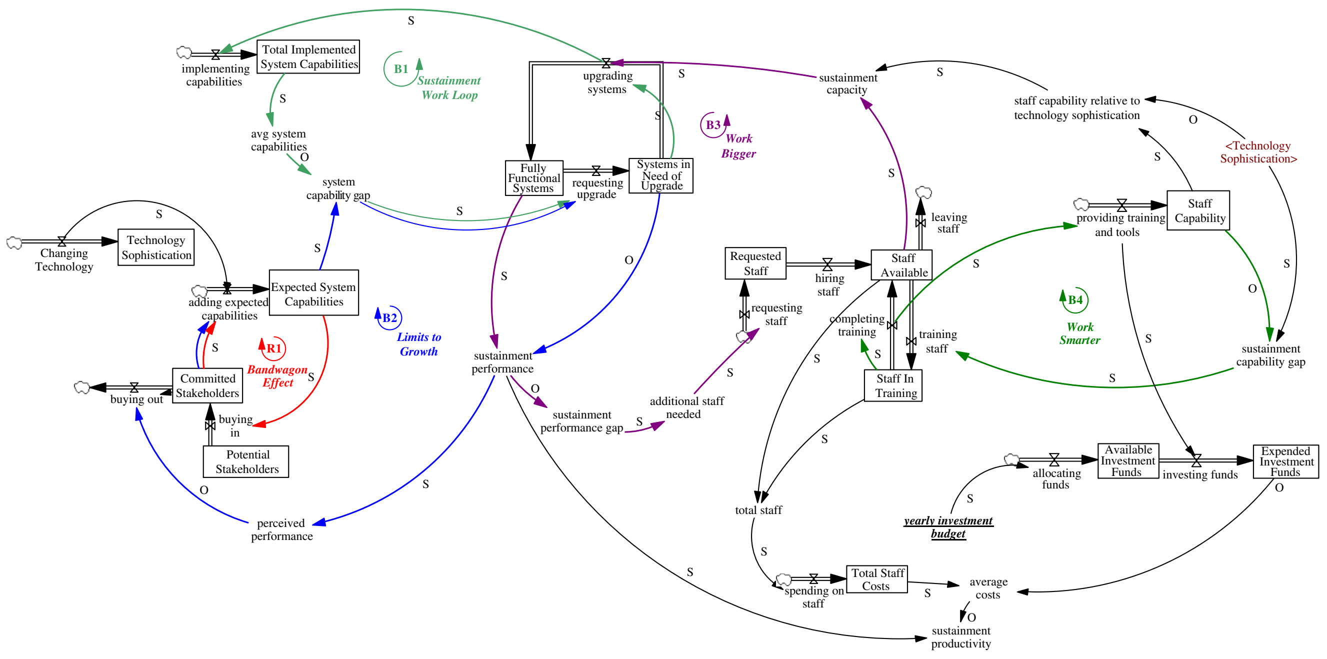

Fig. 5 System dynamics of sustainment.

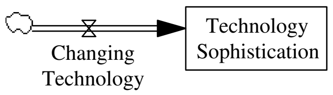

Fig. 6 Example 1.

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169

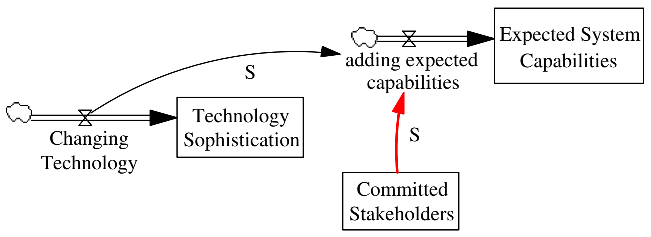
Changing Technology
Technology Sophistication
S
adding expected capabilities
Expected System Capabilities
S
Committed Stakeholders

Fig. 7 Example 2.

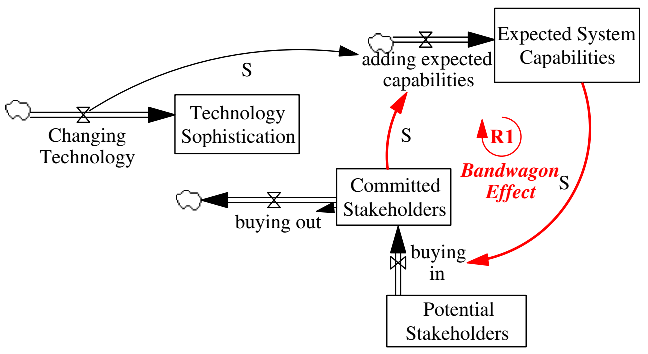

Fig. 8 Reinforcing Loop 1): bandwagon effect.

Figure 10 shows the concepts of sustainment capability and capacity. Increasing technology sophistication reduces the relative capability of the staff: they can still do the job they always did, but their abilities get stale with respect to new technology unless they take some time to learn about it, and perhaps obtain appropriate tools. The reduced staff capability reduces the sustainment capacity of the organization.

Sustainment performance is shown on the left side of Fig. 10. To measure sustainment performance, this model calculates the fraction of total systems that are fully functional. Reduction of capacity slows sustainment work and allows systems in need of upgrade to build up, thereby reducing sustainment performance.

E. Limits to Growth

Figure 11 shows the balancing loop called limits to growth (B2) ([7] p. 379). This loop keeps in check the otherwise ever-increasing reinforcing loop (R1; bandwagon effect). As expectations for capabilities rise, the gap between expected and actual capabilities increases, which then results in upgrade requests. This decreases the number of systems considered "fully functional" and increases the number considered "in need of upgrade." Because the sustainment capacity is limited, systems are not immediately returned to being fully functional, so the sustainment performance suffers. This reduces the number of potential customers buying in and, in fact, can turn the valve the other way, causing previously committed customers to drop out. As the number of stakeholders goes down, their expected capabilities also decrease, thus balancing out the original rise.

F. Work Bigger

Figure 12 introduces the work bigger loop, which shows addition of staff to improve sustainment performance.

If the sustainment performance (calculated as the percentage of fully functional systems) decreases, the gap between actual and desired sustainment performance increases. In response, the organization requests and hires new staff. This involves significant delay, so alternatives such as reallocation of staff from other programs (less delay) or required overtime (least delay) are often implemented, although they are not shown

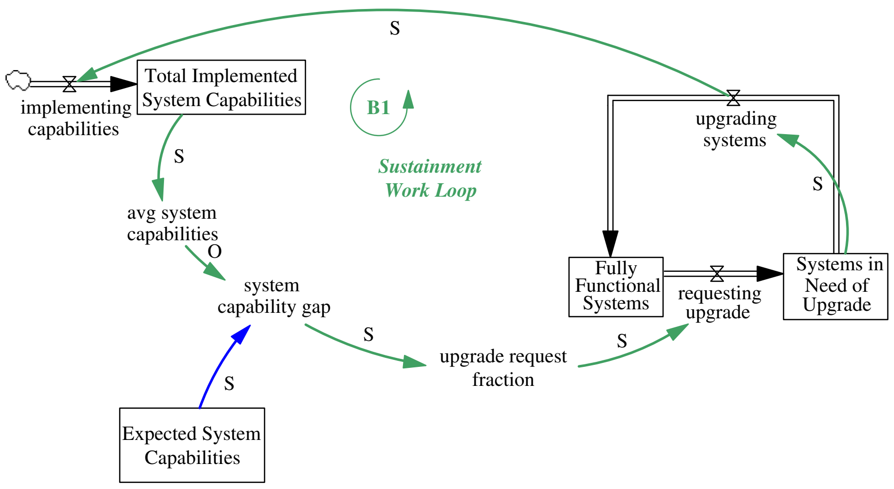

Fig. 9 Sustainment work balancing loop.

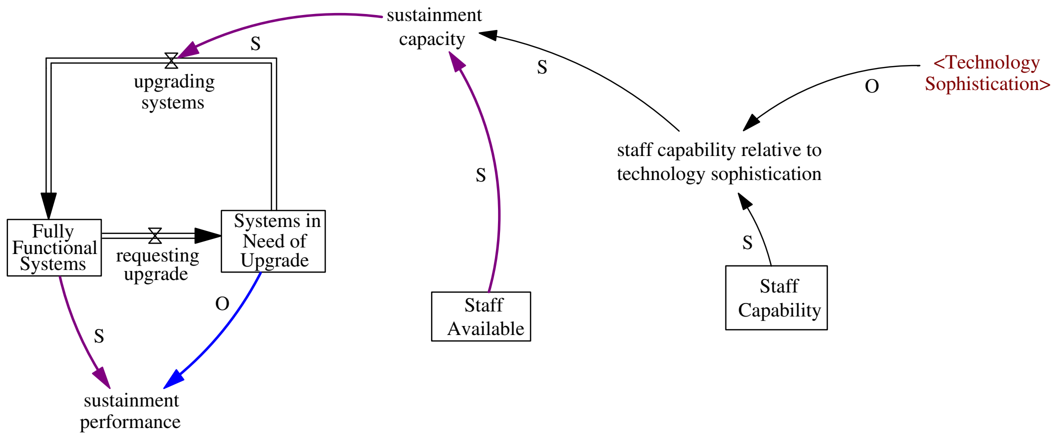

Fig. 10 Sustainment capacity, capability, and performance.

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169

explicitly here. The increase in staff produces more sustainment capacity. This starts to work off the number of systems awaiting upgrade and repair the sustainment performance number. This is work bigger, a balancing loop (B3).

Staff loss occurs at all times, adversely affecting sustainment capacity if hiring does not keep up. Loss is increased if employee satisfaction goes down (not shown on diagram), which can happen if funding problems that prevent hiring add late hours to existing staff, add pressure, reduce training, or prevent the purchase of modern tools. Sequestration and required furloughs, with their lost or postponed paychecks, can exacerbate employee dissatisfaction.

G. Work Smarter

In contrast with working bigger, working smarter involves investing in staff capability. This is shown in Fig. 13. Both staff training and the provision of modern tools increase the staff capability, repairing the gap caused by changing technology. Of course, once the gap is reduced, the need for training and tools goes down and people get back to work. This is the work smarter loop (B4).

The timeframes in this loop are important. When staff are in training, whether in organized classes or just taking down time to learn about a new technology or get used to a new tool, the number of staff available to do work decreases, and thus the sustainment capacity goes down briefly. The increase in capability due to training occurs more slowly, as training needs to be developed or purchased, and courses must wait until people can be taken off the sustainment line to take them.

This results in an immediate decrease, and only a slow increase, in sustainment capacity. This is explained well by Repenning and Sterman [9], who note that it pressures managers to go for the short-term solution (work bigger, our take on Repenning and Sterman's work harder, which is required overtime) and deal at some later date with the inevitable decline in capability and thus capacity.

H. Investment in Training and Tools

Training and tooling require capital (Fig. 14), which represents an investment in the capability of the sustainment organization. The diagram shows these expenses as converting available to expended investment funds. Yearly budget, and the percent of it that is allocated to capitalization, actually depends on the number of committed stakeholders (not shown).

I. Sustainment Productivity

Cost performance is of vital interest to sustainment stakeholders. In this model, sustainment productivity is used as the indicator of cost performance (Fig. 15). Productivity is the relationship of sustainment costs to sustainment performance. Costs include expended investment funds and staff salaries. As productivity goes up, the organization is sustaining the fleet more efficiently.

J. Sustainment System Dynamics Simulation Model

The full diagram shows all of these quantities together (Fig. 5). This is a slightly abstracted causal-loop-style model. A full simulation model was created from this causal-loop diagram that includes almost twice as many variables (including all the auxiliary variables implicit in the

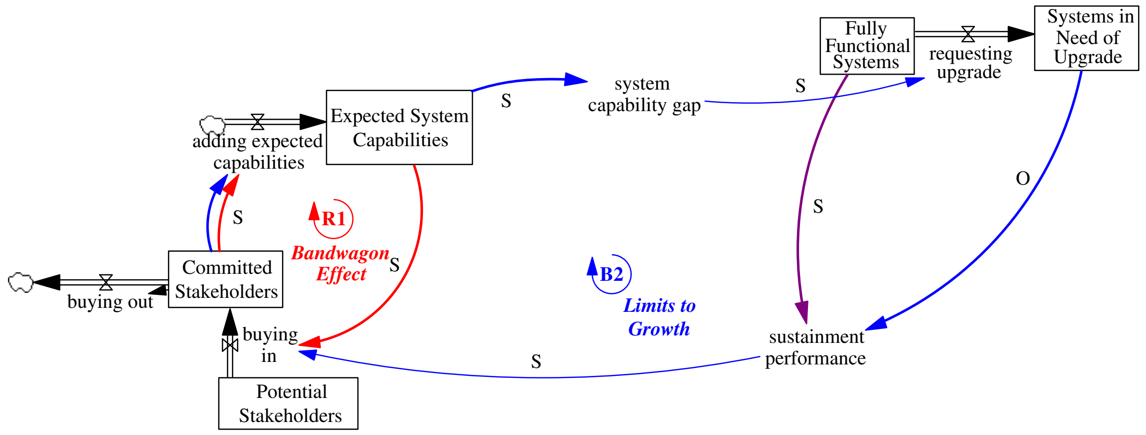

Fig. 11 Limits to growth balancing loop.

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169

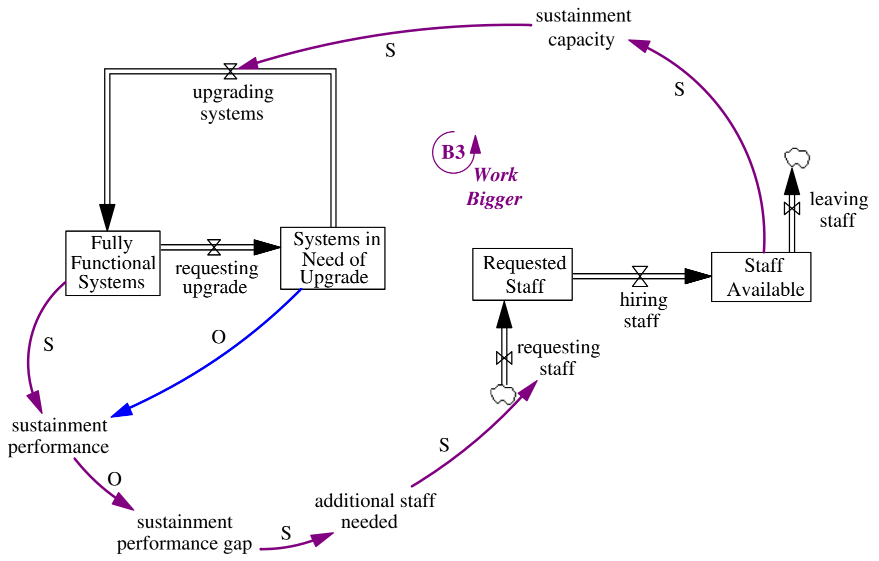

Fig. 12 Work bigger balancing loop.

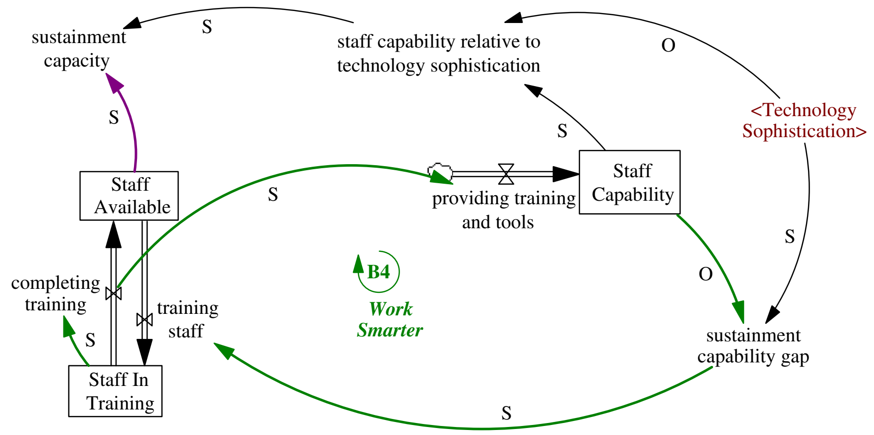
sustainment capacity
S
staff capability relative to technology sophistication
O
<Technology Sophistication>
S
S
Staff Available
S
providing training and tools
Staff Capability
completing training
training staff
B4
Work Smarter
O
S
S
Staff In Training
sustainment capability gap
S

Fig. 13 Work smarter balancing loop.

equations for every parameter). Notional values of all variables have been inserted, and dynamic stability has been achieved. This is an important first step in creating a simulation from a causal-loop diagram. The next step is to run the simulation while inserting changes at a given point. An example of such a change could be to ramp up the amount of technology change (on the left side) while allowing varying amounts of training to take place.

Figure 16 shows what happens when the model is executed in response to a 20% increase in the threat that the system faces, at month 6 for a period of 12 months. This figure was generated using the Vensim® modeling and simulation tool. This behavior-over-time graph shows four simulation runs, indicated by by numerical label (1–4). The simulation runs vary key parameters. The output, on the *Y* axis, is "sustainment performance," which in this case is modeled as a fraction that can vary from 0 to 1.

A baseline simulation (shown as simulation run 1) is run to show steady-state values before introducing a disturbing stimulus: the increased threat. The threat change is modeled as a technology pulse at month 6 during the simulation; for example, an enemy figures out how to counter one

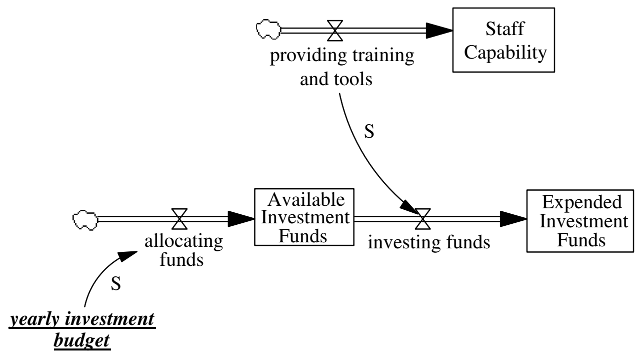
providing training and tools
Staff Capability
S
Available Investment Funds
allocating funds
investing funds
Expended Investment Funds
S
yearly investment budget

Fig. 14 Investment in training and tools.

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169

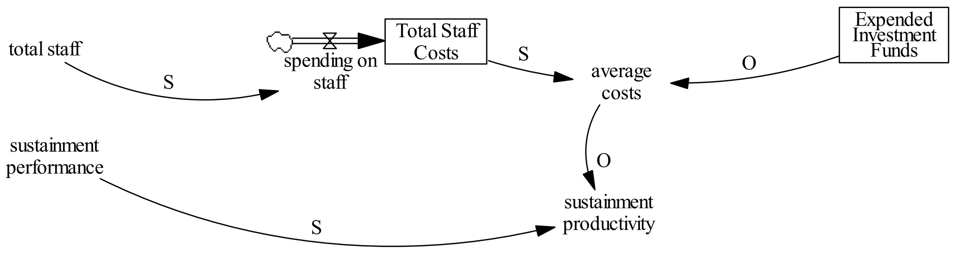

Fig. 15 Sustainment productivity.

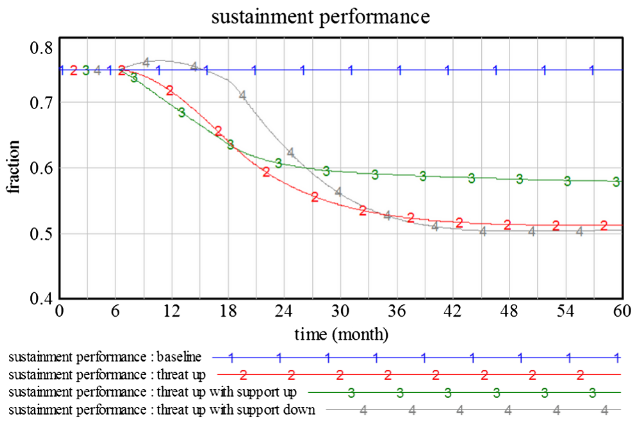

Fig. 16 Sustainment performance with increased threat under varying levels of training/tooling support.

of our sensor or weapon technologies, and a more advanced technology needs to be rolled out to the fleet. The result depends on the ability of the sustainment organization to perform training and provides tools that support sustainment.

The baseline equilibrium (simulation run 1) results in a sustainment performance of 0.75. Simulation run 2 applied a 20% increase in threat at month 6, resulting in performance changes that reach a new equilibrium of sustainment performance at about 0.52, after about three years. Simulation run 3 shows that a higher equilibrium can be established at about the 0.58 level by increasing the level of training and tooling during this high-threat period. Because training takes away from active sustainment work, increasing training (simulation run 3) initially causes lower performance than maintaining the training/tooling status quo (simulation run 2) or decreasing the level of training and tooling during this period (simulation run 4). However, in the long run, increased training pays off better.

Each simulation run makes assumptions about what is held constant. These simulation runs assume that there is no increase in or loss of personnel during the run. An improved outcome could result from increasing hiring, training, and tooling when there is a threat change. The lesson to learn from this simulation is that decreasing the level of training and tooling to get more sustainment hours in the short term sacrifices organizational performance a little later.

Future calibration will help determine the model's validity for sustainment organizations. A system dynamics tool attends to internal consistency, but it does not guarantee that the output has any link with reality. The researchers are experienced, with an average engineering history of over 25 years, so there is a sense of qualitative applicability, but the real proof will be in using an organization's own data and predicting some of

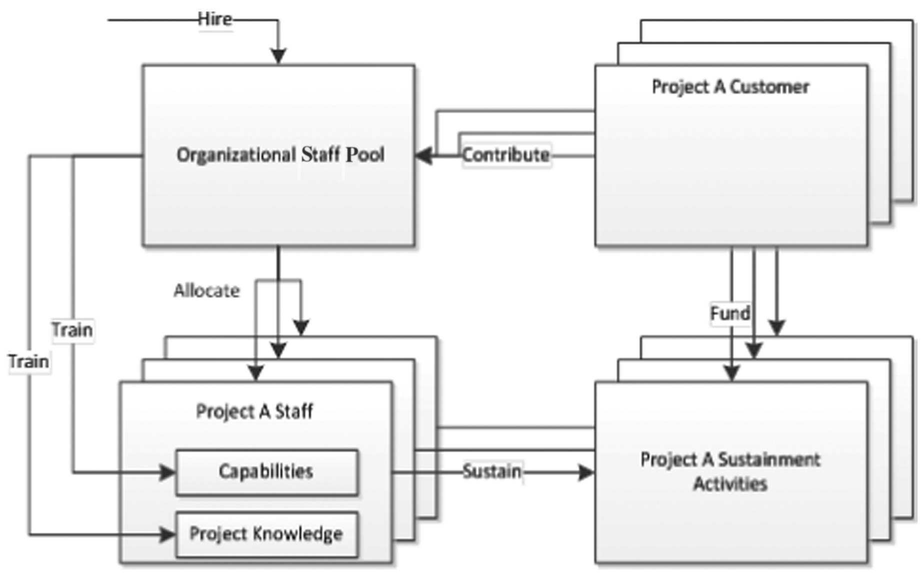

Fig. 17 Portfolio view.

the variables given others, in a variety of situations. Face validity will be demonstrated by collaborating during and after the modeling effort with experts in sustainment organizations.

VI. Future Directions

Potential directions for this research include accounting for a portfolio of projects, different types of personnel, organic vs contractor sustainment, early prediction of sustainment costs, early preparation for sustainment activities, and automating input into the model.

The first of these potential directions, accounting for the portfolio nature of a sustainment organization's work, would involve investigating how different systems or programs interact and how the organization invests in such factors as a centralized sustainment capability, including tools and processes. Figure 17 illustrates how the portfolio model might be set up. It shows a sustainment organization that is sustaining products from three different customers. The organization hires the staff and assigns the appropriate number and skill sets to each product. Each product's sustainment activity operates like the current system simulation model. However, the portfolio view also considers that, when a new technology affects one product, the organization may be able to reassign staff currently working on a different project. In addition, the organization has to plan training for its entire staff pool rather than planning training only within each project. This enhancement to the model was requested by several of the government organizations addressed for this study.

Customers have suggested that the model could be useful in comparing the economic effects of using organic (government employee) personnel to do sustainment work, hiring contractor personnel managed by government managers, or contracting the entire sustainment job to a contractor. This would involve studying howthe different services account for contractor and organic personnel and modeling the intersection and union of their practices.

Questions of predicting sustainment costs of a program early have arisen and require some study to determine the utility of such an approach. Can the architecture suggest sustainment costs, or would it be foolish to estimate before the design is known? Similarly, are there aspects of the design that have been shown to require higher or lower sustainment and, if so, can the cost of sustainment be estimated using a checklist of sustainment impacts to different design features?

Also, has the development phase adequately prepared for sustainment? To what extent were the artifacts created in development phase still usable in the sustainment phase? How many sustainment skills, tools, or facilities were required by the design and code, and could these have been reduced by judicious focus on sustainment needs during development? Creation of a model to help assess these impacts would be useful to many acquisition and development efforts.

A notional cost account structure has been derived by the U.S. Army.[§] It would be useful to examine how well that structure can be used to provide automatic data input into this system dynamics model. Can instrumentation of a sustainment organization following this cost account model help predict sustainment dynamics?

Finally, sensitivity studies should be performed to determine how sensitive any of the aforementioned conclusions are to changes in input.

VII. Conclusions

System dynamics provides a way to simulate the complex set of interactions that underlie sustainment and provides feedback on how changes in some variables affect others. The simulations done to date demonstrate some of the expected interactions among variables that have been noted in real sustainment situations. Next steps include obtaining feedback and organizational data from subject-matter experts in collaborating organizations to calibrate and validate the model. Ultimately, the measure of success of such a model is that it will be used to assist decision-making.

Acknowledgments

This material is based upon work funded and supported by the U.S. Department of Defense under contract FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. This Carnegie Mellon University and Software Engineering Institute material is furnished on an "as-is" basis. Carnegie Mellon University makes no warranties of any kind, either expressed or implied, as to any matter including, but not limited to, warranty of fitness for purpose or merchantability, exclusivity, or results obtained from use of the material. Carnegie Mellon University does not make any warranty of any kind with respect to freedom from patent, trademark, or copyright infringement.

References

[1] *Guidelines for Successful Acquisition and Management of Software-Intensive Systems: Weapon Systems, Command and Control Systems, Management Information Systems, Condensed Version 4.0*, U.S. Air Force Software Technology Support Center, Feb. 2003.

[2] *Examination of the U.S. Air Force's Aircraft Sustainment Needs in the Future and Its Strategy to Meet Those Needs*, National Research Council, National Academies Press, Washington, D.C., 2011.

[3] Estep, L., "Air Force Sustainment Perspective," March 2012, http://www.ncms.org/wp-content/NCMS_files/CTMA/Symposium2012/Plenary/0950Estep .pdf [retrieved 15 April 2013].

[4] Dörner, D., *The Logic of Failure: Why Things Go Wrong and What We Can Do to Make Them Right*, Metropolitan Books, New York, 1996, pp. 185–200.

[5] Lientz, B., and Swanson, E., *Software Maintenance Management*, Addison Wesley, Reading, MA, 1980, pp. 492–497.

[6] "Limitations on the Performance of Depot-Level Maintenance of Materiel," U.S. Code 2466 of Title 10, 2009.

[7] Senge, P. M., *The Fifth Discipline: The Art & Practice of the Learning Organization*, Doubleday, New York, March 2006.

[8] Sterman, J. D., *Business Dynamics: Systems Thinking and Modeling for a Complex World*, Irwin/McGraw–Hill, New York, 2000, pp. 191–195.

[9] Repenning, N. P., and Sterman, J. D., "Nobody Ever Gets Credit for Fixing Problems That Never Happened," *California Management Review*, Vol. 43, No. 4, 2001, pp. 64–88.
doi:10.2307/41166101

L. Long
Associate Editor

[§]Jim, J., McGarry, J., Poland, J., and Jones, C., "Software Maintenance, Sustaining Engineering, and Operational Support: Army Software Maintenance WBS Version 4.4a," U.S. Army Office of the Deputy Assistant Secretary of the Army for Cost and Economics/U.S. Army RDECOM-ARDEC Quality Engineering and System Assurance, Fort Belvoir, VA/Picatinny Arsenal, NJ, 21 February 2013.

Downloaded by CARNEGIE MELLON UNIVERSITY on October 28, 2015 | http://arc.aiaa.org | DOI: 10.2514/1.I010169