2006 CCRTS, THE STATE OF THE ART AND THE STATE OF THE PRACTICE

Title:

Two Theories of Process Design for Information Superiority: *Smart Pull* vs. *Smart Push*

Topics (in descending order of apparent relevance):

C2 Architecture, Policy, Network-Centric Metrics, C2 Concepts and Organizations, C2 Analysis, Cognitive Domain Issues

Author information:

Rick Hayes-Roth
Professor, Information Sciences Dept., Naval Postgraduate School
589 Dyer Road, Bldg. 235
Root Hall 223, Monterey, CA 93943
Telephone: 831-656-3983 or 650-327-1166
Fax: 208-445-0127

ABSTRACT

This paper asks how information should flow among networked entities in NCOW. In particular, should the entities *actively seek*, acquire and process relevant information or should they *wait to react* to information that others send to them? In short, should they *pull* information or should they rely upon others to *push* information to them? In most tactical contexts, *smart push* will improve efficiency by orders of magnitude compared to *smart pull.* Our analysis reveals that efficient information processing chains require a general capability to watch for key events. Humans and the computer applications supporting them will use this capability to detect events matching *conditions of interest* they specify. This capability plays a key role in transforming networks into integrated value chains. Where traditional networks aim at supporting unregulated exchanges for data bit flows best suited to random access and unpredictable process sequences, the capability to delegate condition monitoring enables us to transform networks into conveyers of *timely, valuable information*. To maximize efficiency, we must use processes where each successive step receives just such valuable information as its input. Thus, condition monitoring and its associated smart push constitute a required foundation for the efficient process chains needed to achieve information superiority.

BACKGROUND

Two basic alternatives exist for providing needed inputs to process steps, whether we are discussing supply chains of material goods or information processing chains associated with decision-making and control. Processing entities[1] can seek out relevant inputs and, upon finding them, procure them. Or the entities can inform suppliers of their requirements and depend on the suppliers to deliver needed inputs at the right time and place. A major shift occurred in manufacturing over the last two decades associated with "just in time" (JIT) approaches and "supply chain integration." Top-performing enterprises shifted from the former process design approach to the latter one. When suppliers have an excellent understanding of their customers' processes, schedules, and ongoing process state, they can deliver valued inputs just when they are needed. In manufacturing, this reduces costs in significant ways, especially by reducing inventory and work in process (WIP) that consume space, time, and processing resources associated with storing, searching, and moving around those items not immediately required by the next processing stage. Thus, inappropriate or low quality inputs, as well as inputs received at an inopportune time, increase costs and may actually represent *negative* value.

Systems designed to produce quality decisions have many parallels with manufacturing systems, though the former work by adding value to "bits" where the latter add value by transforming "molecules." The key to high performance, in both cases, is to produce the most valued products as efficiently as possible. Value reflects the degree to which the products embody superior features and qualities and get to "market" promptly. Efficiency, on the other hand, means producing these valued products with a minimal consumption of resources. To achieve that efficiency we use the best tools within well-designed processes. Usually, we improve efficiency by eliminating as many sources of friction as possible that consume resources unnecessarily or increase latency. Friction may be physical or virtual, as when differences in information systems introduce delays and difficulties in accomplishing successive steps.

In moving to network-centric operations and warfare (NCOW), the Department of Defense (DoD) has recognized the importance of reducing friction that makes sharing of information between different entities difficult. When information isn't represented in a standardized way, an entity that wants to find it, procure it, and apply it can incur major delays and difficulties. So, DoD is moving to make all information readily locatable, readable, and interpretable (Wolfowitz, 2004). It hopes to establish consensual meta-data schemas to accomplish this. Even if this approach works, it leaves the question of information *logistics* still unanswered: how can the information supply chain be integrated most efficiently? In particular, should decision-makers seek out and *pull* the information they need, or should suppliers *push* the right information to them at the right place and at the right time?

[1] Some entities that make decisions and perform actions are human and others are information-processing machines. When the military performers are humans, we often call them *operators.* Frequently, we'll just use the abstract term "processing entity" to ignore the nature of the process performer.

The current approach to the design of the DoD Global Information Grid (GIG) and its Network-Centric Enterprise Services (NCES) emphasizes *pull*. The work being done by my colleagues at the World-Wide Consortium for the Grid (W2COG) and me, on the other hand, focuses on the alternative approach. In our approach, networks are designed to optimize information logistics by implementing what we call *Valuable Information at the Right Time* (VIRT). In this approach, suppliers work with intelligent computing machinery to determine which bits should flow to which consumers, thereby integrating the information supply chain in a manner parallel to the recent advances in manufacturing.

This paper aims to clarify the two alternatives and expose the conditions under which each provides a superior information logistics solution.

PROPOSED APPROACHES

The information logistics[2] problem is concerned with optimizing the flow of bits to processing entities distributed across a network. We could formalize this, but it's probably most useful to provide an informal, intuitive characterization of the problem. We consider any number of information producers, who can operate on various inputs to produce information outputs and, in addition, any number of decision-makers that convert selected inputs into outputs that correspond to choices. Some of those choices might trigger actions, through coupling to effectors. Other choices become information inputs to additional processing entities. Some of the entities are people and some are machines or software programs running on machines.

In all systems various limitations constrain attainable results. In most of the distributed systems we are concerned with, constraints limit how much can be done, how well, and how quickly. If we are trying to conduct a battle or minimize casualties from a natural disaster, we typically have too few people, too little time, too little information, and too little available computing resources. Different systems in different contexts will experience these constraints in different orders, but the constraint induced by limited human processing resources typifies crisis response situations. The information logistics challenge is to optimize the quality of results obtained by such a system when resource constraints limit its effectiveness. The essential question facing system designers is <u>how should we select and sequence information for each processing entity to maximize the value of our results</u>?

Very complex systems can't be analyzed and optimized using a closed form approach. We must rely upon heuristics to guide our design in such cases. One obvious heuristic to embrace in systems of the sort being considered is as follows: "Whenever possible, favor behaviors that make better decisions faster." Equivalently, we want to design processes that improve the quality of decisions and increase the speed of decision-making.

For our analysis, we suppose that valued results are produced through the application of systematic processes. For example, manufacturers make many different mixes and

[2] *Distributed information logistics* is introduced and explained in Chow, *et al.* (2000).

different products as a result of each product instance flowing through a set of process steps specified for that type of product. In a similar way, we view information processing systems as producing information products. These products include decisions and associated actions, resulting from a series of process steps applied to appropriate inputs. A number of steps are performed by people, and others are accomplished by computerized agents. Regardless of the type of processing entity, each step transforms input information into output information. The quality of the output can be assessed along various attributes. The single best measure, where we can obtain it, would be the *value* of the information. The value of an interim result corresponds to the expected improvement in eventual final products attributable to it. If final results improve because some interim decision was reached, the increase attributable to that decision is its value. While difficult to measure precisely, estimation of *value added* is a routine part of all mature supply chains. Processing steps and interim results that don't improve expected outcomes may have zero or negative value, because they consume resources without producing benefit.

In short, information logistics addresses the question of how information should flow among processing entities to optimize the value attained. While no definitive optimum may be obtainable, heuristics employed in supply chain integration seem applicable to the flow of information in NCOW chains. Some of these heuristics are listed below:

Desirable Behaviors

- Keep your most specialized, expensive processing entities busy
- Minimize lateness on the most valuable, time-sensitive products
- Drop low-value products before sacrificing high-value ones
- Minimize product quality problems that ensue from low-quality inputs
- Minimize time lost to set-up and cut-over required to handle changes in products or changes in inputs

Undesirable Behaviors

- Load your processing entities with more inputs than they can process
- Allow some of your required processing entities to wait for needed inputs
- Make processing entities filter out low-quality inputs
- Make processing entities prioritize backlogged tasks
- Make processing entities adapt to input variability

Thus, the basic strategy in achieving optimum product output is to allocate as much of your resources as possible to adding value, working on the highest value tasks where possible. Actions or interim results that waste time, waste processing resources, or produce little or no value should be avoided. While this may not achieve a provable optimum result, systems that adhere to these heuristics clearly outperform those that do not. As a consequence, we will prefer the *better* systems to the *poorer* ones, moving up so-called Pareto frontiers as high as possible.

In the simplest, most extreme form, there are two alternative approaches to the question of managing information flow in these distributed NCOW systems:

- **Theory 1**: Describe all information available using some type of meta-data description. Give each processing entity good search tools. Have each entity seek and acquire whatever information it needs, when and as needed.
- **Theory 2:** Have each processing entity describe conditions that would make its current plans undesirable, because those conditions would contradict assumptions needed to justify the choice of the affected plan. Enable agents to alert the affected entity. Enable the alerted entity to respond quickly to the received news.

In my classes, I have referred to these two approaches simply as **Theory 1** and **Theory 2**. Here, however, we might benefit from recognizing that the first is actually a "pull" approach to information logistics, whereas the second is a "push" approach. In each case, when the tools and computing software exploit domain semantics, ontologies, rules of inference and similar elements of artificial intelligence to improve understanding and performance, we can readily refer to these as "smart," "intelligent" or "knowledge-based." So when an operator in a Theory 1 system can ask for all recent reports about the geographic quadrant he currently occupies, the query processor can employ much knowledge and intelligent reasoning. Similarly, if an operator posts a planned route of flight, altitude, and expected waypoint times, an agent that filters spatially and temporally relevant pilot reports would illustrate a degree of intelligent push.

So both approaches can simplify the work imposed upon a human operator by enabling that operator to express queries in terms of high-level semantic concepts and domain-relevant conditions of interest. As a consequence, our analysis will mostly ignore any questions of whether or how knowledge-based intelligent processing might be performed. Another path we won't explore concerns various ways to mix the two approaches. Instead, we want to focus on the relevant strengths and weaknesses of pull *v.* push approaches to the information logistics question. Our purpose is to develop a crisp appreciation of the two pure and opposite management strategies.

ANALYSIS

Theory 1 is mostly consistent with current US architecture and management directives. Its proponents have been motivated by several observations. First, the US has notorious difficulty sharing information in a timely way, especially from intelligence sources to diverse military and homeland defense operators (National Commission, 2004). Significant backlogs of intelligence observations often sit for long periods before analysts can work their way through them. As a result, intelligence data often don't get to people while they're still current and valuable. The former DoD CIO, John Stenbit, recommended a new information logistics process that posts data as soon as they're available, even before analysts could interpret them and convert "data" into "information" (Ackerman, 2003). Further, former Deputy Secretary of Defense Paul Wolfowitz issued a directive mandating that all DOD entities possessing potentially useful data should describe those data using an appropriate XML meta-data schema as a first step toward enabling all operators to find and access those data (Wolfowitz, 2004). These steps have been leveraged by the DoD National Information Infrastructure (NII)

plans for the Global Information Grid and Network-Centric Enterprise Services (Stenbit, 2004). In their thinking, information superiority will be achieved by providing excellent search tools so that each operator can find and acquire information needed.

Theory 2 was implicit in some of my earlier papers, such as [Hayes-Roth, 2004a, 2004b, 2005a]. It represents a strategy for exploiting the exponential increase in the processing capacity of machines to enable humans with fixed, finite capacities to benefit from increasingly vast quantities of relevant data. Theory 2 aims to mitigate what humans experience as the "data glut." Seeing the similarity between manufacturing supply chains and information decision chains also motivates us to ask how we can emulate the efficiencies of just-in-time processes with lean, low-inventory processes. The challenge in achieving this kind of low-friction, efficient integration in manufacturing has been to make suppliers adapt their products and methods to optimize their customers' performance. This requires that suppliers get smart about how their customers use the suppliers' products and how users' costs can be minimized. As a simple example, FedEx discovered that customers needed to schedule and track shipping from their own premises, so FedEx moved those capabilities to the customers' workstations and adapted their own systems, schedules, and services so the customers themselves could view and control them. Thus, while shipping services were outsourced, each customer actually brought more control of adapted shipping capabilities into their own processes.

The VIRT methodology assumes information supply chains are restructured in a manner akin to such manufacturing supply chain adaptations. In particular, it assumes that suppliers of information learn what customers actually need to know and provide just that information to them. Intelligent agents, playing the role of information brokers, accept statements from processing entities or operators describing "conditions of interest" (COIs). These conditions describe potential events that would motivate the operators to change their planned actions to achieve better outcomes.

As an example, consider a helicopter pilot intending to fly a particular route in hostile territory. During planning, the pilot plots a low risk route. Subsequently, new observations about anti-aircraft emplacements along the planned route of flight would match one of the pilot's COIs.

In short, Theory 2 assumes many processing entities have a continuous need to know about things that undercut their previous decisions by violating some assumptions on which those decisions depend. Operators engaged in real-time plan execution consider such information *vital* and *urgent*. Those attributes, significance and timeliness, make the information high-value. Theory 2 sees information logistics as the challenge of (1) getting high-value information quickly to operators who are dependent on it and (2) assuring operators give priority to received high-value information so they adapt and achieve better outcomes.

Figures 1 and 2 below diagram simple functional models of the value chains associated with the two different types of processing organizations. These models use a lot of informal shorthand to make them simple and easy to read. The first model is centered on

the Processing Entities $PE_1,\ldots, PE_k$ that do work. The PEs add value by accessing various information sources $IS_1,\ldots, IS_n$ to produce valued products labeled *v.* Each PE acquires its inputs through interaction with a Query Specifier (QS). The function *q* on the link between the PE and QS represents the transaction that yields information products from QS with various levels of efficiency. So, for example, we can assume *q* gives a lot of information at low cost, as most query processing systems do. Google, for example, gives thousands of relevant answers to most queries.

The rest of the process works roughly as follows. Once a query is specified, the transaction *p* translates the query into a query plan by working with the Query Planner (QP). The transaction *p* just passes back the responses to the query through QS and then through *q.* The Query Planner uses various Information Directories (ID_j) to understand what kinds of information are available and how to access them. Information Stores (IS_n) store, manage and access discrete bodies of information. The processes used by QP are labeled *r* and *s,* representing the transactions that seek and retrieve relevant information needed to answer the query.

Simple model of Theory 1

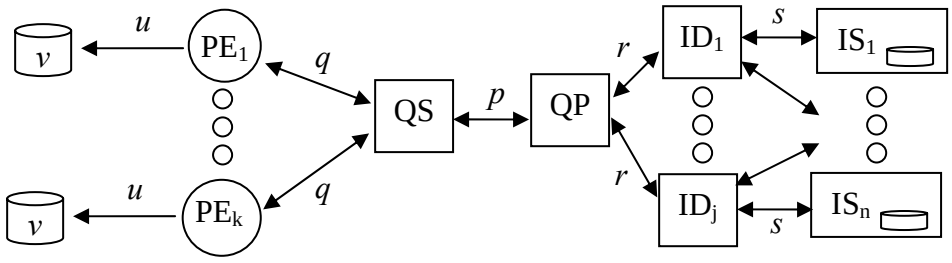

Figure 1. A value chain of processing entities PE_i producing products *v* as a result of specifying queries and planning and executing those queries through information directories to various information sources.

Simple model of Theory 2

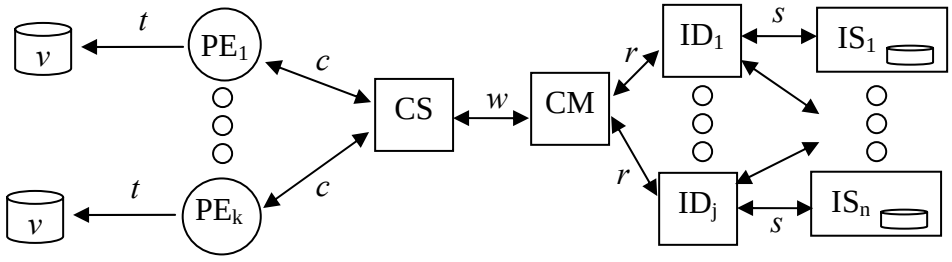

Figure 2. A value chain of processing entities PE_i producing products *v* as a result of specifying and monitoring COIs and then reacting adaptively to alerts.

The second model is very similar, and it too focuses on the same Processing Entities $PE_1,\ldots, PE_k$ that add value by accessing various information sources $IS_1,\ldots, IS_n$ to produce valued products labeled *v.* In this model, however, VIRT processes are at work,

enabling each PE to inform the system about the COIs the system should continuously monitor. Each PE conveys its needs through interaction with a Condition Specifier (CS). The function *c* on the link between the PE and CS represents the transaction that yields information products consistent with PE's specification. So for example, we can assume *c* gives a minimal amount of information at low cost, because the PE specifies precisely what type of events it must be concerned with.

The rest of the process works roughly as follows. Once a condition is specified, the CS conveys it to the Condition Monitor (CM) through *w,* and CM takes responsibility for monitoring it. The transaction *w* just passes back any new events matching the condition through CS and then through *c.* The Condition Monitor uses various Information Directories (ID_j) to understand what kinds of information are available and how to access them. Information Stores (IS_n) store, manage and access discrete bodies of information. The processes used by CM are labeled *r* and *s,* representing the transactions that seek and retrieve relevant information.

Let's compare the two models. While Condition Monitoring differs a bit internally from Query Planning, the two functions use information directories and information sources in nearly identical ways. Thus the real differences between Models 1 and 2 are in the efficiency of Model 1's *u*, *q*, and *p* in comparison to Model 2's *t*, *c*, and *w*. In fact, we can simplify the analysis by viewing the valued products in each case as the output of composing the corresponding functions, so that:

In Model 1, $v = u \circ q \circ p\ (K)$,
and in Model 2, $v = t \circ c \circ w\ (K)$,
where *K* denotes all available information and meta-data .

We read these informal equations to say, in the case of the first model for example, the value produced (*v*) equals what *u* extracts from what *q* extracts from what *p* extracts from all available information and meta-data. Equivalently, *p* finds relevant information in *K* and passes it to the next process step; this step applies *q* to find and pass along relevant information; finally, the process step *u* finds and identifies the valued information *v.*

For our analysis, we're assuming the valued products *v* are the same under the two models. These would correspond to identified threats requiring a helicopter pilot to divert from planned course, for example, or other "needles in the haystack" that operators would find worthy of selecting to act upon. In such a context, then, the two models produce the same end result, but Model 1 forces the Processing Entity to consider through process *u* vastly more inputs that result from query *q.* In Model 2, on the other hand, the Processing Entity has defined a precise COI that would materially affect expected outcomes. This means that the Processing Entity can mostly pass through results of *c,* eliminating any need for *t* to perform filtering and prioritization.

Let's consider a quantitative example. "Threats" that might affect a helicopter pilot can be natural or man-made in origin. The natural category would include high terrain, poor visibility, excessive winds, thunderstorms, or icing. Of course, these only affect the pilot if they intersect the helicopter's route of flight. Man-made threats include ground-based anti-aircraft weapons, fixed or mobile surveillance assets, and enemy aircraft. These may

pose a threat if the helicopter's route intersects the volume of space these systems can observe or reach. When we speak of "intersecting," we mean the threat occupies the same space at the same time as the helicopter. So, to compare our two models, let's look at a helicopter pilot who's concerned about these possible threats intersecting the planned route of flight.

The terrain phenomena are relatively static, so there's little value in considering terrain data repeatedly, unless the pilot changes the route of flight some time after planning it. In that case, terrain previously considered benign may become a threat. On the other hand, weather changes constantly so weather data are worth considering continuously. Similarly, enemy assets may move around and allied intelligence may also improve its estimate of their positions and capabilities. For this reason, information about enemy capabilities deserves continuous consideration.

So how do the two alternative models address these needs? Each needs to look for *relevant* information and determine if it's *significant*. Information about the different phenomena is represented in various ways, but a good simplification is that all phenomena are modeled as values of appropriate variables stored in some gridded geospatially indexed array. For example, *winds aloft* are reported in terms of direction and speed at each latitude-longitude grid coordinate, at each of several corresponding altitudes (such as 3000', 6000', 9000', etc.) The coarseness of the grid mesh differs for different types of variables in different types of systems. Usually, the geospatial data in one data array represent the expected values of the corresponding variable at a particular time. Data values that are forecast for different points in the future are stored in different arrays, each corresponding to one forecast time. When users need values that are intermediate between grid points or time points, they normally interpolate between adjacent values of the variable of interest.

In a small square-shaped theater of operations that might measure 200 km on a side, let's consider how much data is available. Let's assume that all grid meshes are 1 km, so that we have 40,000 grid points for each variable corresponding to each pair of latitude and longitude grid coordinates. We'll assume that in the vertical dimension we have data for every 500 m, from 0 (sea level) up to 6 km altitude, for a total of 13 altitude coordinates. Thus, for each variable of interest, for each moment in time, we have about 500K data values. Let's consider missions that last 4.5 hrs for which we have forecast values for each 30 min, so initially we have 10 distinct time coordinates. This means that initially, our data universe is 5M values for each variable of interest. We'll assume that we want to monitor just 10 variables in total, so that means 50 M values form our base of potentially relevant variable values. Normally, data are updated, based on new information and estimates, at least twice per hour, which means that every thirty minutes all of the data might potentially change. Because changes occur mostly asynchronously, the best strategy is to revisit the data of interest periodically, to be able to notice and respond quickly to important changes. Let's assume that our pilot decides to reexamine data every 10 minutes throughout a 4.5 hr mission.

Model 1 suggests, then, that the pilot should retrieve all relevant data every 10 minutes. The data of interest are all values of the 10 variables that intersect the planned route of flight, in the sense of overlapping spatial volume of capability with the helicopter, at the same time[3]. For simplification, we'll assume that 10% of the total data universe corresponds to the points in space and time where interactions might occur, if the variables indicate a threat such as a thunderstorm or enemy aircraft. So every 10 minutes, the pilot's queries access and retrieve through processes *r* 10% of 50M values, perhaps further reducing them by 90% using filters in *q* to exclude insignificant items from all the relevant data values retrieved. This means that about 1% of 50M values are returned to the pilot, or 500 K relevant and significant data values[4]. Usually, a tiny fraction of these will justify a change in plan. Most of these values, in fact, won't make a significant difference in expected outcome for the pilot. In addition, the pilot's own process *u* will be overtaxed by this volume of data queued for human processing. As a result, most of the data will be ignored, reactions will be suboptimal, and the pilot will feel continually stressed by the repeated onslaught of data deluges arriving every 10 min.

In contrast, Model 2 suggests that the processing system should take responsibility for monitoring COIs that would probably motivate and justify adaptive responses. These conditions correspond to changes in expected values of variables that the plan depends on for its success. So, for example, if there's no risk associated with faster than expected ground speeds that could result from favorable tail winds, there's no reason to monitor for tail winds. As another example, if fuel reserves are provided for 90 minutes beyond expected flight duration and are allowed to go as low as 45 minutes in all cases, only strong headwinds are worthy of considering. In fact, the system could compute the expected impact of headwinds on total flight time and only alert the pilot when the threshold of 45 minutes fuel reserve is in jeopardy. As another example, data on enemy capabilities that confirm previously considered information are insignificant to the pilot, because the system is looking for *new* threats.

Model 2 exploits its awareness of the pilot's prior and current knowledge, in addition to the pilot's plan, to drastically reduce the information passed to the pilot. The model allows the pilot to convey that understanding through process *c* where the pilot formulates specific COIs. Simplifying a bit, the PE asks the system to notify it of changes in expected values, *i.e*. events, which negate or make questionable the ability of the PE to perform its mission successfully. In the current example, over each 10 minute interval the pilot will probably receive zero or a very small number of alerts stating that some variable values have changed in significant ways. The pilot, in Model 2, has plenty of

[3] The planned route occupies a series of 3D points over time. The data of interest describe variable values at 3D points over time. The two sources are intersected over the 4D coordinates of space-time.

[4] In today's best systems human processing capabilities are optimized by presenting much geospatially indexed data graphically as multi-colored maps. This allows the human perceptual system to process much data in parallel and detect interesting phenomena visually. We're ignoring special processing capabilities of particular machinery so we can focus on the more essential question of how to assure high-value information flows and represents an extremely high proportion of all communicated bits. All of the human's processing capabilities are limited, and they should not be squandered processing low-value inputs.

mental cycles available to handle these rare and important alerts. Furthermore, the low rate of data avoids stressing the pilot, and that further enhances human performance.

So Model 2 reduces the volume of information being communicated and also reduces the amount of work that the PE has to perform. This makes a PE in Model 2 much more efficient than in Model 1. In our example, Model 2 reduces the input volume to the PE by a factor of more than 100,000 (five orders of magnitude). If the PE in Model 1 had huge processing capacity, it could perhaps produce all desired valuable outputs *v,* just as well as in Model 2. Model 1 requires applying the "needle-in-the-haystack-finding" process *u* to about 500K items, every 10 minutes. This requires *u* to operate on 3M items suggested by the smart, excellent query *p* every hour. Typically, however, the PE is resource limited, because PE is a human and has limited cognitive bandwidth. Suppose 10% of the pilot's mental capacity is available for this task, and that humans can consider 10 significant variable values a minute. In an hour, the pilot could consider 600 reported relevant variable values, or just $1/50^{th}$ of 1 % of all the retrieved information. Obviously, the pilot's response will reflect a rather random or arbitrary selection of the relevant information.

Model 2, however, requires that the pilot consider a small number of items, almost always well below the cognitive processing limits. Model 2 achieves this efficiency by conveying to condition monitoring agents a lot of "context" about the operator. For example, the Condition Monitor can take a set of assumed routes, way points, and required meteorological conditions for each pilot and continually compare and contrast the forecast weather with the required conditions. Only when weather degrades relative to a particular pilot's requirements for a specific route at a specific time and place would the CM need to transmit data back to the PE. In contrast, a system built around Theory 1 would require the pilot to ask for weather information periodically, to determine which returned values had changed significantly and, finally, to calculate which had degraded to a point where the expectable mission outcome would no longer be acceptable.[5] Theory 2 suggests that all of that work should be avoided, and Model 2 shows that it can be avoided at low cost by delegating context-aware condition monitoring, as performed by CM.

Are there any situations where Model 1 is as efficient as Model 2 or even better? Yes there are. Model 2's superiority depends upon an ability to delegate to an agent the responsibility for monitoring all key assumptions expressed in terms of COIs. This

[5] In this analysis, to make Model 1 as efficient as possible, we've assumed that the queries could be personalized and contextualized so that only relevant values of variables would be provided to the pilot. Thus, we've assumed in implementing Theory 1, the information universe would be honed to a small number of variables of interest and, further, that the values returned would be restricted to those where forecast values in space-time intersect with planned space-time coordinates for the flight. These assumptions go well beyond what actual systems offer or what is even being anticipated among Theory 1 practitioners. However, such improvements are logically independent of the particular Theory. We are incorporating these improvements by assumption in Model 1 to make any claims for Theory 2's advantages more vulnerable to rejection. The primary weather information systems for pilots, as just one example, afford pilots *none* of these beneficial improvements. By imagining Theory 1 is made as efficient as possible, we avoid any bias in an analysis which ultimately favors Theory 2.

requires an understanding of how outcomes depend on various variables and conditions. It also depends on capabilities to monitor those variables and compute those conditions. Furthermore, the monitoring system must be informed when plans and underlying assumptions are changed. All of this works best in contexts where plans are produced and maintained in digital, symbolic systems, and where planning processes make explicit rationales supporting decisions and choices. The military has many operations where these conditions apply, as in most tactical planning, maneuver, and logistics. However, even in these cases, the formalization of semantic concepts (ontologies), conditions of interest, pertinent information directories and information sources has not progressed very far.

So Model 1 seems to be a better fit for activities where people are engaged in less structured tasks, as when they are trying to become familiar with a new area, trying to build an initial understanding of a new situation, or when they are looking for ill defined patterns to emerge from massive amounts of data. In such situations, data mining algorithms and human intuition and perception can often be superior. Browsing, in the absence of some clear notion of what one is looking for, seems not to be a function where machines have much competitive capability.

Internet search engines, such as Google, have shown that Model 1 can be extremely valuable, especially for a wide variety of users who collectively have overlapping interests. By suggesting to the next person who asks the answers that previous investigators tended to value, search engines offer considerable advantages in comparison to human, manual, or unordered search. Model 1 seems to make concrete the idea that there are Processing Entities, usually human, that value shot-gun answers to general questions.

In contrast, Model 2 promises enormous increases in productivity for operations that are engaged in performing defined processes, repeatedly, where resources are limited and outcomes are important. In such cases, we may routinely need to let some potential opportunities “hit the floor,” and there’s great value in assuring effective prosecution of the most important objectives on time and within budget. In these contexts, we want our people attending only to important issues. We do not want them spending time scanning, browsing, filtering, and prioritizing incoming queues that are overflowing with relevant but mostly insignificant information, or managing growing backlogs of unfinished tasks. To maximize their productivity, we want their input queues to be automatically prioritized continually so that the next item they process is, in fact, the one that has the highest expected value added. Model 2 makes that the routine process management approach.

MONITORING CONDITIONS OF INTEREST

When bandwidth is limited, we want to use it wisely. In any given system, bandwidth can measure the maximum possible rate of information flow through some component. Usually, of course, we think of communications bandwidth, a measure of the capacity to transport bits from one place to another per unit time. In other systems, we think of CPU

clock rate or memory access rates, measures of the capacity to access or manipulate bits per unit time. As the preceding example illustrates, many processes depend on human thinking to succeed, but people can only consider a small number of variables or questions per unit time.

All systems that process information exhibit an upper limit on the amount of useful work they can accomplish per unit time, because at some level of workload, one of their components hits its maximum rate of throughput. We refer to the component that limits the total system at that point as "the rate-limiting component" or "the gating factor." After we exceed the capacity of the rate-limiting component, we incur other problems and costs. If inputs continue to arrive, they must either be stored in some temporary buffer, housed permanently in a more expensive or slower facility or, as often actually happens, they simply "hit the floor." In that case, the inputs are damaged or lost, never to be recovered.

When interacting with a dynamic environment or a quick enemy, it's vital that our systems effectively focus on the most important inputs so they can implement adaptive responses before the environment or opponent creates the next problem requiring response. The basic idea in intelligent control is that one's rate of considering and acting must be faster than the opponent's. So, even when our systems have limited capacity, they must give priority to important information and assure their basic adaptive cycle is quick enough (Hayes-Roth, 2005c). As a result, many items may "hit the floor" when we act efficiently. However, we can't afford to let chance determine which items the system ignores. Even when we have limited resources, we must attend to and respond to the most significant events. In our example above, the pilot in Model 1 couldn't do this, because the deluge of inputs exceeded the resource capacity to recognize and act on the most significant information.

To integrate efficient supply chains, suppliers learn what their customers need and how they should supply inputs to reduce the customers' difficulty, cost, and delay (*friction)* in employing them. In our pursuit of NCOW and its corresponding information chain integration, we want to apply the same principles. This encourages us to focus on the basic function of supplying information to "customers." By analogy to the supply chain, each processing entity should supply information in a manner that minimizes the friction incurred by the intended recipient. In addition, because most information chains will be rate-limited, every processing entity will need to attend first to the most important bits. If suppliers can prioritize information for them, processing entities can spend their limited resources more effectively, assuring they process important items before unimportant ones. In this way, the entire system can maximize the resources it expends processing the most vital bits.

To support this basic objective, we want the system to assure that processing entities receive prioritized information in a form that simplifies the recipient's task of understanding it and reacting to it adaptively. This is what Model 2 does by allowing each processing entity to specify its own conditions of interest. Each COI corresponds to a potential event that undercuts the expectation of a successful process or mission. One broad class of COIs logically corresponds to the negation of a prerequisite. For example,

sufficient fuel is a prerequisite for flight. Each planned route considers initial fuel, fuel consumption per hour, and total hours of flight, for example. When people consider, analyze and then select a plan, they assess fuel consumption and *justify* the plan by their calculations showing the aircraft will have *sufficient fuel* to execute the plan. The corresponding most general COI is "expected fuel consumption exceeds amount available (less required reserves)." Another general class of COI corresponds to the failure of an assumed salutary condition. An example of this might be "absence of enemy threat along route of flight." While plans may not absolutely require such a condition, it's obviously desirable. In fact, in considering two alternative routes, the planner may have chosen this specific one precisely because of this very belief. These kinds of COIs reflect the importance of recognizing and adapting to events that increase the risks of failure.

A very different category of COI addresses the importance, in some operations, of adapting to increased opportunities, where we see risks as "opportunity costs," rather than chances of failure *per se.* Thus, when forecast thunderstorms dissipate, opportunities arise for shorter, faster, and safer routes.

Any NCOW system should enable processing entities to get the high-value information they need in a form that makes it easy and quick for them to react to it. This can be done by enabling each PE to specify its own COIs, reflecting its own knowledge, expected plans, and concerns about various kinds of risks. The ability to delegate and monitor COIs seems critically important. Systems intended to foster information superiority must provide this essential capability to the human operators and automated processing entities. This is probably **the single greatest way to use computing power to increase process efficiency** and to assure that limited human bandwidth can focus on important information.

Using COI monitoring this way supports the objective of every processing entity receiving *valuable information at the right time* (VIRT). Networks that embrace this process design approach will manage the flow of bits to maximize the value of those bits. We term that approach to network management *flow by value* (Hayes-Roth, 2004b, 2005a).

To enable processing entities to specify their COIs new language systems will be required, and these will constitute improved query languages. Such new language systems will require three principal components:

1. **Domain-specific ontologies**. The concepts, variables, value intervals, and similar attributes required to describe the information needed for effective missions collectively define the domain ontology. As an example, an ontology for the helicopter missions would include concepts such as *turbulence*, *icing*, *anti-aircraft missile*, *vehicle track, ground speed*, and *fuel consumption*.

2. **Domain-specific expressions and conditions**. Expressions specify how to calculate quantities useful in determining whether a significant value exists, as well as particular values signifying important events, *i.e.*, instances of conditions

of interest. Examples would include expected *fuel exhaustion*, expected *airframe icing*, or expected *detection* by a capable anti-aircraft enemy system.

3. **Condition monitors**. Monitors are programs that take responsibility for computing conditions and alerting the interested processing entities that specified them. These monitors will access relevant data, compute the expressions associated with the conditions, and determine when the resulting values warrant notifying the interested entity or operator. In systems with extensive amounts of information, this function will be a heavy consumer of machine computing cycles. (Hayes-Roth & Brown, 2005, specifies an architecture for one such capability.)

Specialized tools to help construct and continually improve these three kinds of components will prove extremely valuable. We might consider such tools three additional types of components, making a total of six important types of capabilities required to make COI monitoring routinely available within NCOW.

Beyond the technology, organizations will need to adapt their processes as well. In the military, tactics and procedures should evolve to make COI monitoring a routine process suitable for automation. The military already has a related concept termed the Commander's Critical Information Requirements (CCIRs). Loosely speaking, these define conditions whose occurrence justifies "waking the Commander up." In today's practice, Commanders delegate CCIRs to human staff officers, and so these COIs are expressed informally in natural language.

In order to make every processing entity efficient, we look forward to the development of excellent formal systems that enable people to state their COIs precisely and simply in terms of familiar domain ontologies. In most cases, the humans should use interactive language tools that allow the computer to construct unambiguous formal interpretations while, at the same time, simplifying human performance requirements. Once formulated, the COIs would be routinely delegated to computing machines that would monitor them reliably.

We might expect these COIs to be termed something like *Dynamic Operator Information Requirements* (DOIRs). It seems straightforward to adopt a practice of explicitly defining DOIRs as a part of the planning process. DOIRs would be expressed in terms of the ontology and expressions appropriate to each domain of operation. Because all organizations spend most of their time repeatedly performing standard processes, most organizations should create a catalog of frequently used DOIRs. This would make it quick and easy to specify DOIRs for today's mission, which often would be just a slight parametric variation of items in the organization's catalog.

DISCUSSION

I've shared earlier drafts of this paper with a number of colleagues throughout government, industry and academia. They have raised a number of points that I'd like to mention and consider here. There are four specific points that we'll consider in turn, and these are: (1) the best architecture will need to support both smart push **and** smart pull;

(2) *push* and *pull* may be misleading or harmful terms; (3) people shouldn't depend on automation to provide the information they need; and (4) smart information flows require better semantics than our systems possess.

Point 1: The best architecture will need to support both smart push **and** smart pull. I've tried to emphasize to this point in the paper the significant advantage of smart push over smart pull. The earlier example showed an advantage of more than five orders of magnitude in terms of reducing information to what's essential. This is the single greatest quantitative advantage in efficiency that I've encountered in my IT career. This advantage isn't yet widely understood, and many people have a poorly informed negative bias toward smart push, mostly stemming from early Internet products that deluged enterprises frequently pushing graphical data to users. Moreover, most of the GIG/NCES and DoD policy documents emphasize access so users can seek and pull relevant data. Collectively these conditions make it important to argue against the idea that smart pull is *the* solution. Not only is it not the solution, it's not even a tolerable approach for many operational contexts.

However, as I suggested earlier, there are many contexts where smart push is not the answer either. Many of these contexts involve intelligence gathering or other open-ended investigations. We can be confident there are countless tasks where operators would benefit from an ability to seek and quickly find pertinent information. After all, the most useful function of the Internet today would seem to be search, which is a kind of pull. So, a broad, effective, general IT architecture will need to support both push and pull. My intent is to put smart push on the agenda and to move it to a high-priority position that reflects its enormous advantages in many operational contexts.

Point 2: P*ush* and *pull* may be misleading or harmful terms. Readers have made me aware that these terms have been used in a variety of ways, ranging from technical to marketing communications, with some attendant over-loading and confusion. Hopefully, readers of this paper understand that I've been focusing on the flow of information to recipients, distinguishing *pull* flows that require recipients to periodically look for relevant data from *push* flows that cause relevant data to arrive at the recipient's inbox. Some readers point out that these functions are implemented in various ways, depending on the nature of the distributed infrastructure, and sometimes the implementations share common elements. Many of the elements of the end-to-end flows, as shown in the two process models, are similar, of course.

Nevertheless, I think it's instructive to look at the process models at a high level and consider which method provides more precise information to the operator and which method requires less work by the operator. It should be clear that two things are required to maximize the ratio of value to effort: (a) the operator's dynamic context must be used to determine precisely how current estimates differ from expectation, because those differences constitute the candidate bits of value; and (b) the system must maintain dynamic context and past data to detect changes of interest. In addition, additional benefits accrue if the system also can project future states and detect deviations between prior expectations and now-predicted situations. Because information becomes valuable

by improving an operator's expected outcome, operators will most value bits that correctly alert them to the need to change course to avoid a predictable problem or exploit an emerging opportunity.

In short, there shouldn't be any confusion about the terms, as used in this paper. Smart pull means getting the most relevant information you can when you search for it. Smart push means letting others know your context and relying on them to alert you when they determine you are at risk.

Point 3: People shouldn't depend on automation to provide the information they need. Automation is often considered dangerous or harmful, because people may become over-reliant upon it. When automated systems fail or when novel problems arise beyond the system's boundaries, we need people to take over and provide agility and robustness. In this paper, for example, a reader might worry that our operators will become over-dependent on systems to tell them what they should know. One consequence, the argument goes, is that we weaken our operators as a result. Smart push lulls them into a kind of false level of comfort and excessive dependency.

I don't think there's anything specific to IT or smart push in such concerns. All automation ultimately creates dependencies and, in fact, people lose the skills previously required to do those functions used prior to automation. We don't have many skillful slide-rule operators today, as handheld calculators and spreadsheet software have made it possible for everybody to do complex mathematical computations at the push of a button. Few people can write database queries or Boolean information retrieval queries, yet everybody can seek and find more data, more easily, with Internet search engines. Pilots of complex aircraft regularly rely on GPS for navigation, autopilots for flying, and flight management systems to determine routing and fuel management decisions. In short, technology makes us more productive precisely because it eliminates tasks that can and often do become obsolete.

With respect to the specific question of whether smart push can provide productivity advantages, there's no doubt. Will this necessarily mean operators lose the ability to find relevant information when they want it? Will it mean that operators will no longer be effective when their systems break or are unable to address new contexts? We are certainly a very long way from having to address those questions, because we have scarcely begun to address the question of how we'd give operators just the high-value information they'd want in various dynamic contexts. At this low baseline level on the productivity scale, with little technology assistance, we know that operators can't get the information they need easily and quickly by any means. It seems obvious then that the preponderance of risk is associated with poor performance due to poor information. We should look forward to an eventual problem of having extremely high performance due to automation that provides precise, high-value information. Before operators become overly dependent and vulnerable in that eventual state of information superiority, we can introduce back up measures to increase robustness and maintain essential human skills.

Point 4: Smart information flows require better semantics than our systems possess. The examples in this paper show how smart push or pull can find relevant information and compute conditions of interest. To do that, operators define COIs using expressions that computers can evaluate. The expressions or queries use a vocabulary of entities or variables, attributes, and values, and often these are indexed with spatial and temporal coordinates. Finally, these values might be indexed by additional aspects corresponding to different perspectives, hypothetical cases, or other context-indicating factors. A *schema* for all these data defines what kinds of bits are stored and how they're organized.

To match data to COIs, we must recognize the correspondence between the operators' concerns and the type of information modeled in the schema. In short, we need to have alignment in meaning between operators and the people who supply bits. Semantics is the term we use to refer to such meaning. As any system designer or data engineer knows, getting agreement on meaning is tough work, and the requirements continue to evolve as missions and contexts change.

So smart information flows will depend upon our machines incorporating semantic models suitable for the tasks we apply them to. The semantic models will underlie the tools users employ to express their COIs and to view the alerts and events the system pushes to them. Information will need to be registered against appropriate coordinate systems and other reference frameworks for the semantics to be meaningful and correct. Furthermore, information might be supplied using one semantic framework, compared to information in another semantic framework, and ultimately presented to operators in a third, context-sensitive framework. All of this suggests that semantics will be important, and a continuing area of learning and improvement.

There are many efforts underway throughout industry and government to create improved semantic frameworks, including ontologies and tools for comparing and translating between different category systems. I don't expect this work to move especially quickly or reach completion in the foreseeable future. On the other hand, we don't need complete solutions to begin to deliver great value to operators. We only need enough semantic foundations to understand COIs in specific operational contexts and correctly monitor evolving data relevant to those COIs. In this way, we can deliver value to operators "one thread at-a-time," by implementing the semantics-enabled processes across just those information sources needed to monitor those COIs. If we plan for continuous improvement and evolution of the semantic frameworks and semantics-supported processes, we should be able to achieve great productivity gains for the indefinite future.

CONCLUSIONS

When we don't really know what information we need or how it will change our behavior, we have few means of achieving great efficiency. In such a case we have little choice but to browse all potentially relevant information sources in the hope of noticing something interesting and, even better, something significant with respect to our situation or planned actions. This type of information system, consistent with Model 1, places a great burden on processing entities. They must seek information broadly, they must have

broad access, and they must acquire large volumes of generally relevant information. Then they must process it, interpret it, look for significance, filter and prioritize it. In a network-centric operation, we should expect such processing entities to have long latencies, to induce bottlenecks, and to exhibit large backlogs of interim products awaiting further processing. Some functions probably must be based on Model 1, such as some general intelligence operations, but we should expect efficient organizations to have few such processes.

On the other hand, we should expect mature, efficient organizations to have well integrated value chains, where each processing entity allocates most of its time to near immediate adaptive responses to *exceptions,* events that don't jibe with expectations and that jeopardize desired outcomes still in process. Efficient systems strive to achieve the best possible results by using reliable processes, each step adding value in predictable ways, and all operating within an envelope of *nominal* conditions. These systems spend most of their resources adding value by building in predictability through methods that minimize variance, reduce friction, and prevent the off-nominal unexpected. In the rare cases where conditions go off-nominal and the expected consequences are negative, the systems quickly notice them, because they are actively monitoring for just those kinds of expectable problematic conditions. In these systems, significant events occur infrequently. Further, the processing entities have mostly short queues with no backlogs, meaning they have capacity to deal with urgent events. As a result, these systems can give priority immediately to understanding a surprising event, contemplating its negative impact, and considering adaptive responses. Model 2, in short, allocates scarce attentional and problem-solving resources, often dependent on humans, in a near optimal way to recognizing and responding to vital and urgent events.

The theory of NCOW, roughly, predicts that we should be able to accomplish better results by distributing information quickly to everyone who could benefit from it. That seems like a good idea. However, Theory 1 and Theory 2 suggest two complementary strategies for achieving those improved outcomes. This paper shows that most advances in information technology, ranging from increased processing power to formal semantics, advanced query processing and knowledge-based inference, contribute significantly greater productivity gains in the context of integrated supply chains operated according to Theory 2. For this reason, organizations should give priority to designing and implementing processes as Theory 2 suggests.

REFERENCES

Ackerman, R. K. (2003). Network centricity begins at home. *SIGNAL*, August.

Alberts, D., Gartska, J., *et al.* (2000). *Network Centric Warfare: Developing and Leveraging Information Superiority" (2nd ed.).* OSD/ASD (C3I), CCRP, http://www.defenselink.mil/nii/NCW/ncw_0801.pdf .

Chow, Y.-W., F. A. Hayes-Roth, *et al.* (2000). Automatic retrieval of changed files by a network software agent. US Patent & Trademarks Office, #6,029,175.

Hayes-Roth, F. (2003). Architecture, Interoperability, and Information Superiority. NPS Research Paper. Monterey, CA.

Hayes-Roth, F. (2004a). Next-generation battlespace awareness. *Adaptive Information: Improving business through semantic interoperability, grid computing, and enterprise integration.* J. T. Pollock and R. Hodgson. Hoboken, NJ, Wiley-Interscience: 341-342.

Hayes-Roth, F. (2004b). Model-based communication networks for improved collaboration and decision-making. Keynote Address. 12th International Conference on Telecommunication Systems - Modeling and Analysis, NPS, Monterey, DON CIO.

Hayes-Roth, F. (2005a). Model-based Communication Networks and VIRT: Filtering Information by Value to Improve Collaborative Decision-Making. 10th International Command and Control Research and Technology Symposium: The Future of C2, McLean, VA, US Department of Defense, Command and Control Research Program (CCRP).

Hayes-Roth, F. (2005b). Towards a rich semantic model of *Track*: Essential Foundation for Information Sharing. NPS Research Paper. Monterey, CA.

Hayes-Roth, F. (2005c). *Hyper-Beings: How Intelligent Organizations Attain Supremacy through Information Superiority.* (Pre-publication draft.)

Hayes-Roth, F., and D. Amor (2003). *Radical Simplicity: Transforming Computers into Me-Centric Appliances.* New York: Prentice-Hall.

Hayes-Roth, F. and G. Brown (2005). VIRT Technical Architecture specification. W2COG Technical Working Group Document.

Hayes-Roth, F., J. E. Davidson, *et al.* (1991). Frameworks for developing intelligent systems. *IEEE Expert* **6**(3): 30-40.

Lark, J. S., L. D. Erman, *et al.* (1990). Concepts, methods, and languages for building timely intelligent systems. *Real-Time Systems* **2**(1/2): 127-148.

National Commission on Terrorist Attacks (2004). *The 9/11 Commission Report: Final Report of the National Commission on Terrorist Attacks Upon the United States*. New York: Norton.

Oros, Carl (2005). Helicopter Information Awareness Module (I-AM): An example of a Model-Based Communication Network (MCN) Architecture. 10th International Command and Control Research and Technology Symposium: The Future of C2, McLean, VA, US Department of Defense, Command and Control Research Program (CCRP).

Stenbit, J. P. (2004). Statement before the Committee on Armed Services, U.S. House of Representatives, Subcommittee on Terrorism, Unconventional Threats and Capabilities. February 11. http://www.house.gov/hasc/openingstatementsandpressreleases/108thcongress/04-02-11stenbit.html

US DOD/NII TRM (Technical Reference Model) & Global Information Grid Architectures. http://trm.disa.mil/ and https://disain.disa.mil/ncow.html

Wolfowitz, P. (2004). Data Sharing in a Net-Centric Department of Defense. Department of Defense, Directive 8320.2 (December 2004), ASD (NII)/DoD CIO,