12th ICCRTS
"Adapting C2 to the 21st Century"

Title of Paper:

Mapping Network Centric Operational Architectures to C2 and Software Architectures

Topics
C2 Technologies and Systems; C2 Metrics and Assessment; C2 Concepts, Theory, and Policy

Authors: Jack Lenahan, Imagine-One Corporation
Phil Charles, Command Chief Engineer,
Space and Naval Warfare Systems Center Charleston, South Carolina
Rebecca Reed, SRC Corporation, Don Pacetti, ManTech Corporation,
Mike Nash, SPAWAR Charleston
POC: Jack Lenahan
Organization: Office of the Chief Engineer
Space and NAVAL Warfare Systems Command
Charleston, S.C.
Address: P.O. Box 190022
N. Charleston, South Carolina: 29419
Phone: 843-218-6080
Email: John.Lenahan@Navy.mil

Abstract

We are interested in mapping Operational Architectures to Command and Control and Software Architectures. In his poignant paper, Dekker[1] proposed a "Taxonomy of Architectures" which provides an interesting spectrum of network centric operational configurations. The goal of this paper is to answer the following question: given the variety of architecture models presented by Dekker, what command and control model is appropriate for each, and what software architecture is appropriate for each? In particular, this research will examine if the German control free model applies to all of the different operational architectures and whether or not the service oriented architecture (SOA) is the appropriate software architecture solution for each of the models. Our gedanken experiment results show that the configuration of assets and how they were organized (commanded and controlled) actually increased their collective capabilities given an optimized hybrid SOA, MOMS (Message Oriented Middleware), and Agent Based software infrastructure. This means that any capability portfolio analysis or competency assessments which only focuses upon individual asset contributions, fails to account for the behavior of a team or the possibility of "collective swarm intelligence". This almost by definition will lead to procurement decisions detrimental to the basic capability of the DoD.

Introduction

This paper describes the results of a gedanken[2] experiment. Thought experiment methodology is a priori, rather than empirical, in that it does not proceed by observation or physical experiment. Thought experiments are well-structured hypothetical questions that employ "What if?" reasoning. In our case, we wish to evaluate the proposed architectural taxonomies of Dekker as a possible set of operational baseline configurations with respect to their relationships to command and control models (C2) and software architectures. The Dekker architecture types which will be evaluated are hub request, hub swarming[3], request based (without a hub), emergent swarming (leaderless), hierarchical swarming, orchestrated swarming, and distributed swarming (leaderless). For this gedanken experiment, a swarm is assumed to have the properties of swarm intelligence normally associated with Particle Swarm Optimization[4] (PSO). Due to time constraints, Dekker's Joint, and Mixed models were not evaluated. The command and control models[5] evaluated were: *cyclic* (where senior headquarters issues orders to all subordinates, but does so on the basis of a preset cycle time. The Chinese Army and the Soviet World War II forces adopted this approach because their communications structures could not provide continuous information to the central headquarters and because their subordinate organizations were unable to display initiative in the absence of detailed directives); *interventionist* (relies heavily on central authority to issue directives, but also maintained very detailed information about the battle (requiring continuous and specific reports from subordinates two layers down) and attempted centralized control through detailed directives, mainly used in Soviet style structures); *control free* (seeks to assign missions to their subordinates, who are then expected to employ all the assets available to them to accomplish the missions. This requires a military organization where the lower echelons are competent and trusted implicitly by the higher echelons. The system designed by the Germans for World War II is the case that fits most clearly in this category); *selective control (*in which higher headquarters also issue mission-type orders and expect subordinates to take broad and deep initiatives. However, their higher headquarters follow the battle in detail and are prepared to intervene in the event of a major opportunity or major threat that the lower level command does not perceive or cannot manage. This approach requires great discipline on the part of the senior commanders, who have tactical-level information and considerable skill as tactical commanders, but only intervene when operational or strategic level issues emerge. In essence, the Israelis prefer rapid reaction on the battlefield but seek to maintain the capability for central intervention); *problem bounding* (the higher headquarters tend to compose their directives in terms of the objectives to be accomplished, but to couch them in very general terms. Hence, directives are more specific than mere mission assignments and some explicit boundaries (deadlines for achieving some objectives, guidance on risks that might be accepted or avoided, etc.) are articulated. British plans for an operation tend to be less detailed than those of Americans, often by a factor of three to one); and *problem solving* (missions and objectives are articulated for two levels of subordinates and substantial guidance about how the objectives are to be achieved is also included. Although this approach provides more detailed direction than the UK philosophy, considerable room remains for lower-level initiative and creativity in accomplishing the objectives. At the same time, however, the high-technology assets which US forces tend to employ often mean that subordinates are heavily dependent on senior commanders for key assets such as lift, intelligence, supplies, or precision munitions).

The software architecture models examined were: *service oriented architecture* (SOA) which is a software architecture layered upon an existing enabling software infrastructure, which delivers content and services to consumers primarily via the use of Web Services; an *event driven architecture* (EDA) is a software architecture which is centered on the state changes (events) of an automata or database. The EDA is characterized by the initiation of message traffic due to the occurrence of change of a database row or rows through insertion, deletion, or updating. In the case of automaton state changes, events are usually described as program subroutines completing, object oriented methods completing, or the initiation of a request for data by a user; a *legacy software architecture* is software that can be characterized informally as old software that is still performing a useful job for the community; *message oriented middleware* software architecture (MOMS) comprises a category of inter-application communication software that generally relies on asynchronous message-passing as opposed to a request/response metaphor.Most message-oriented middleware depends on a message queue system, although some implementations rely on broadcast or on multicast messaging systems; and Agent Based Architectures (ABA)[6]; Cougaar defines an agent as a software entity which autonomously communicates with other software agents to achieve domain-specific functionality. Multiple agents often collaborate as peers in a Peer-to-Peer (P2P) distributed network. The complexity of each agent can range from simple embedded sensors to a highly complex artificial intelligence application. In the context of this paper, each agent must be able to learn, may contain their own genetic algorithms, learning classifier systems (LCS) or may contain their own neural network topologies.

The full set of definitions of these terms can be found in the glossary at the end of this document.

Evaluation Process

The operational architecture configurations that follow were assessed by subject matter experts (SMES) with some simulation for certain configurations and some actual software architectural testing for others. The conclusions reached in this paper are thus those of the SMES and do not represent exhaustive experimentation of these combined architectures. Again, these are gedanken experiments.
While the majority of the analysis was performed by SMES, it was supplemented by a few simulations wherever possible. The simulations were performed using commercially available software packages and MATLAB. The software architectures were evaluated as follows:

1. The agent based architecture and the leaderless (emergent swarming and distributed swarming) architectures, were simulated using the models presented[7] at the June 2006 CCRTS in San Diego. Genetic algorithms were implemented using the MATLAB GA toolkit to provide machine learning and optimal planning and re-planning

2. Service Oriented Architecture evaluations were constructed in two different methodologies. The first used the NCES model as a basis with various calls for federated queries from the agents. This is how the knowledge to plan or re-plan was provided particularly for the swarm models. The second SOA approach used models of web services acting as API's for calls to legacy planning systems. This amounted to a "hard wired" SOA versus a discoverable set of services. These were evaluated for hub models. The reason for this was to validate any possible overhead introduced by a" pure SOA" in terms of slower response times for certain requests which may or may not be due to identity management or security management layer latencies. However, this is not covered by this paper.
3. The EDA was evaluated by adding DB row insertion time, trigger firing time, and stored procedure execution time to the MOMS formulas below. This was performed by SMES.
4. MOMS were simulated / evaluated using Little's Theorems[8]. The following queuing theory formulas were used: Probability of n queue entries being active at time t = $p_n(t)$, Average number of active queue entries at time t = (Avg)N (t), Average delay (avg T_k) of k^{th} entry in a queue being serviced, $T = \lim_{k \to \infty} \text{Avg}(T)_k$, Little's Theorem $N = \lambda T$, Average number N of active queue entries with the average delay T, Queue Waiting Time - $W = \lambda X^2 / 2(1-p)$
5. Legacy software times were provided by SMES. More validation of those results will be performed later this year.
6. At least 1 hour was assumed for the chain of command permission (problem solving model) requests for a revision of the search plan and the issuance of new orders[9] to the drones.
7. At least 2 hours was assumed for initial plan creation times for the interventionist and cyclic models.

Setting the Context

We begin by defining a simple problem scenario of configuring a set of drones for usage in a mission to find a missing aircraft; we also set the following conditions needed to perform a simple analysis of the proposed operational architectures:

1. Assume that we have a set of drones available on a sensor grid
2. Each of the drones are fueled and available for tasking
3. The drones receive their tasking via GIG Sensor Grid Communications or directly from a "leader drone" in Hub models
4. Assume that there are only 3 types of drones available:
    a. For this simple example, all drone types have equivalent sensors & range
    b. Each drone of the first type contains onboard artificially intelligent software agents capable of planning a search and rescue mission
    c. Each drone of the first type can be appointed as a command node and issues search pattern commands to the non command nodes
    d. Each drone of the second type cannot plan a mission and can only follow orders
    e. Each drone of the third type is used for protection only. Thus it cannot be used in searches in hub type architectures

    f. Depending upon the architectural configuration, the onboard agents will be able to communicate with each other or only to a leader.
    g. Drones of all types can communicate with the sensor Grid or each other
5. The sensor grid contains an adjudication agent which will deconflict concurrent or competing asset requests. This agent was not implemented or its impact on finding the missing plane analyzed for this effort due to staff resource constraints. This is mentioned only to complete the sensor grid description since given a real set of sensors, some task and sensor request adjudicator will be necessary.
6. The metrics which will be used to judge each configuration are:
    a. Time for the leader drone to process the mission request and "understand it", for hub or leader based architectures.
    b. Time for the "swarm" to process a mission request and "understand it" in non-hub models
    c. Time for a leader node to create a search plan.
    d. Time for a swarm to create a search plan.
    e. Time to compute the area of uncertainty by a leader
    f. Time to compute the area of uncertainty by a swarm.
    g. Time to determine the search plan for each individual drone by a leader
    h. Tine to determine the search plan for individual drones if calculated by the swarm
    i. Time for requests to be processed from each drone to the leader
    j. Time to re-plan by a leader model if first searches are unsuccessful
    k. Time to re-plan by a swarm if first searches are unsuccessful
    l. Time from mission start until mission completion (missing plane found)
7. The graphics used will compare legacy to a hybrid combination of an SOA, ABA, and MOMS for the tasks evaluated under varying C2 structures. The other results for individual software architectures are not being presented due to space constraints.

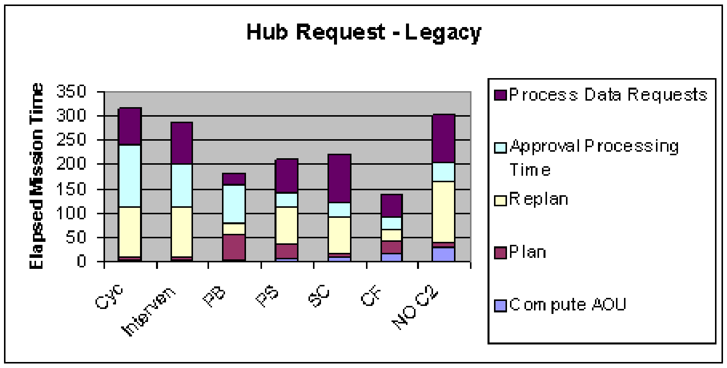
Analysis

We begin with a missing aircraft last seen at a particular latitude and longitude. The task is to find the aircraft within a specified time frame. We now ask the following questions. What is the best operational configuration for the available assets given the search and rescue mission? Will multiple configurations work equally well? What software architecture should be used? And what is the appropriate model of command and control?

Examining configuration 1 – Hub Request

A Hub Request[10] Architecture is a configuration characterized by "a single high-value central "hub" node, surrounded by a cluster of nodes of lower value. The central "hub" provides services of such high value that the force cannot operate effectively without it. The "hub" is therefore what Clausewitz called the "*center of gravity... on which everything depends*". The high-value of the "hub" means that its services will be in high demand, and some method is required to prioritize and balance requests. The potential vulnerability of the "hub" means that it must be protected. But this means that

by definition, one or more subordinate drones must be available for protection and cannot be used in the search in any meaningful degree.
Thus, our first configuration will be a hub model using one of the drones as a centralized command and control "leader" and one drone assigned as protector. The leader drone is of type 1; all the other drones are of type 2 except for the protector drone. Thus we begin this configuration with one less search asset.
After receiving a stimulus, the leader drone after start of mission creates a plan, determines the area of uncertainty for the search pattern, requests the number of servant drones required, receives permission to use the drones, and issues tasking orders to each drone in terms of its own particular search pattern. At this point we will superimpose the six C2 models on this configuration and determine the resulting metrics.



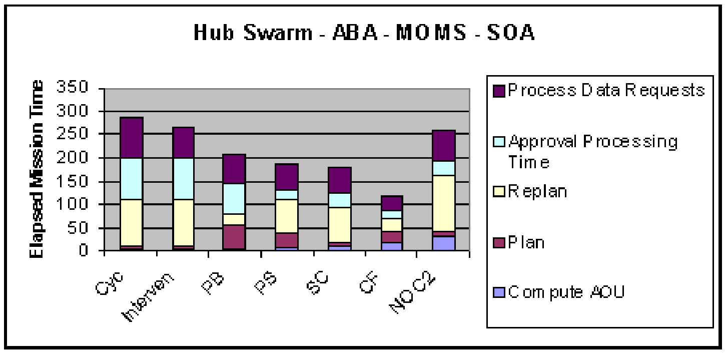
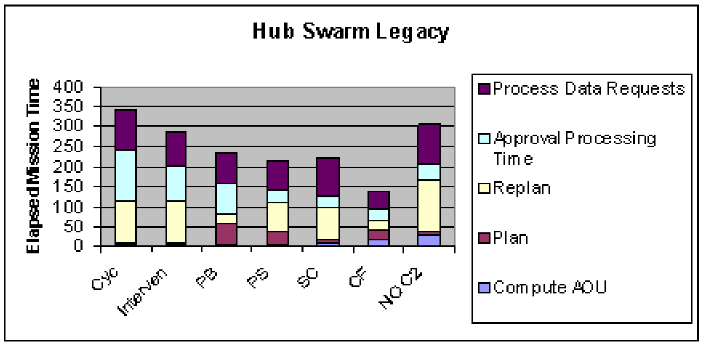


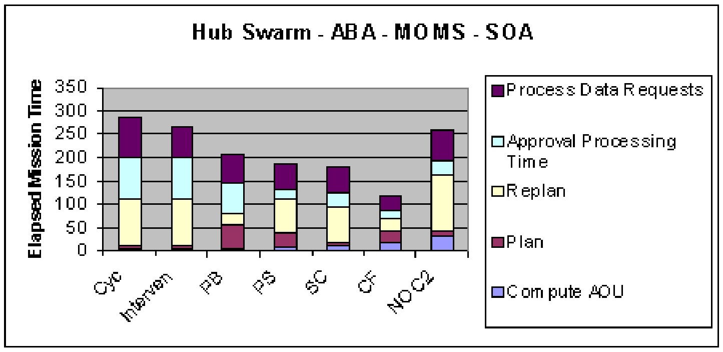
Figure 1 – Hub Request Model Results

Results for the Hub Request using legacy indicate that the minimum total elapse mission time was achieved using control free C2. All C2 models improved with the addition of the SOA for data request processing and the MOMS for approval time processing. Planning and re-planning also improved with the SOA since data was more readily accessible. The "No - C2" model had to re-plan excessively causing the longer elapsed mission times in its case. This occurred because the identical mission was repeated many times. Thus if the plane was not found in the same time frame, we declared the C2 type to have performed in an inconsistent manner.

<u>Examining configuration 2 – Hub Swarm</u>

The second configuration will be a hub swarm model using one of the drones as a centralized command and control "leader". Hub Swarm[11] architectures involve a mix of nodes of different kinds and values. Such a mix arises particularly in a Joint force, and involves mixing elements of all the other types of NCW. How this mixture should be achieved is dependent on understanding fully the other types. High-value nodes within a Joint architecture will behave more or less like "hubs." Groups of similar nodes will to some extent display swarming behaviour. In addition, requests will be passed between different kinds of node. Achieving such a "seamless" Joint force will therefore require exploring the other types of NCW at least at the concept demonstrator
The leader drone is of type 1; all the other drones are of type 2, except for the protector drone. The leader drone after start of mission creates a plan, determines the area of uncertainty for the search pattern, requests the number of servant drones required, receives permission to use the drones, and issues tasking orders to each drone in terms of its own particular search pattern. By SME analysis, the results for this model are depicted in the graphic below.





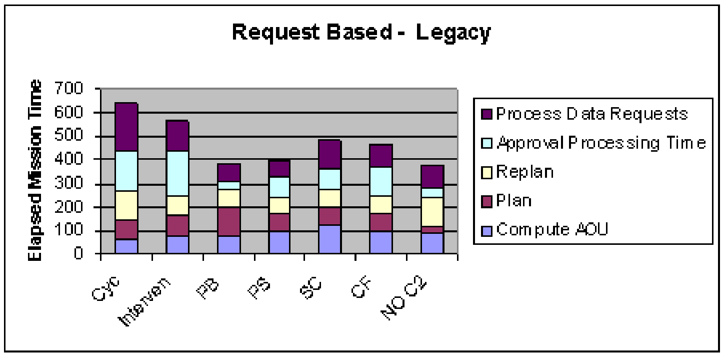
<u>Figure 2 – Hub Swarm Model Results</u>

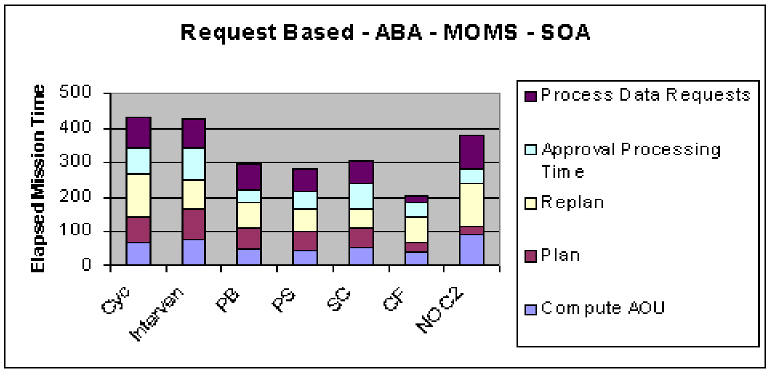
Results for the Hub Swarm Operational Architecture using legacy indicate that the minimum total elapse mission time was achieved again using control free C2. All C2 models improved with the addition of the SOA for data request processing and the

MOMS for approval time processing. Planning and re-planning also improved with the SOA since data was more readily accessible. Particle swarm optimization also contributed significantly to improving this result over the hub request model.

<u>Examining configuration 3 – Request Based</u>

The third configuration will be a request based model. A request based[12] architecture, defined as the combination of fully value-symmetric and heterogeneous forces, is a collection of pure specialists, all different, but all of equal value. Each node does only a few things, and does them extremely well. Since military operations require multiple coordinated tasks, each node must call on many others to perform tasks that it cannot do. In this kind of architecture, requests for services are broadcast across the network, and the network identifies possible nodes which can satisfy the request (Hall *et al* 2004). These nodes in turn may require additional services, thus generating further network traffic. For this exercise this model is limited to the three drone types, leader drones, protector drones, and search drones. We did not have time to explore all possible permutations within this context. By SME analysis, the results for this model are:



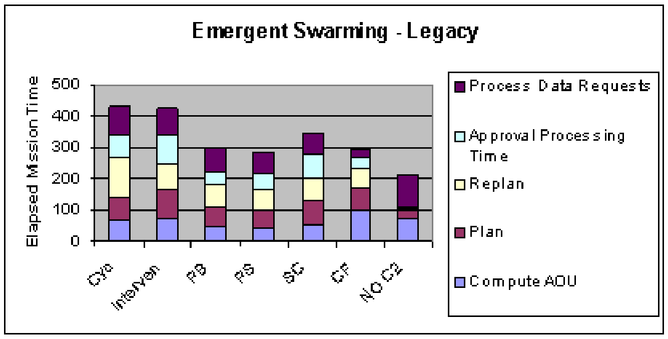


<u>Figure 3 – Request Based Model Results</u>

As Dekker predicted, please note the increased times for additional communications represented in the approval and data request tasks. Also note the same improvement over the legacy software due to improved request processing of the MOMS and SOA style architectures.

Examining configuration 4 – Emergent Swarming

The fourth configuration will be the emergent swarming model. Emergent Swarming[13] occurs in nature among insects such as ants (Gordon 1999): “The basic mystery about ant colonies is that there is no management. A functioning organization with no one in charge is so unlike the way humans operate as to be virtually inconceivable. There is no central control. ... No ant is able to assess the global needs of the colony, or to count how many workers are engaged in each task and decide how many should be allocated differently. The capacities of individuals are limited. Each worker need make only fairly simple decisions.” We shall also assume that the range and sensor capabilities are equal. Each drone must plan for itself. By analysis the emergent model performed many tasks faster but much more unreliably than the other configurations. The summary results for this model are:



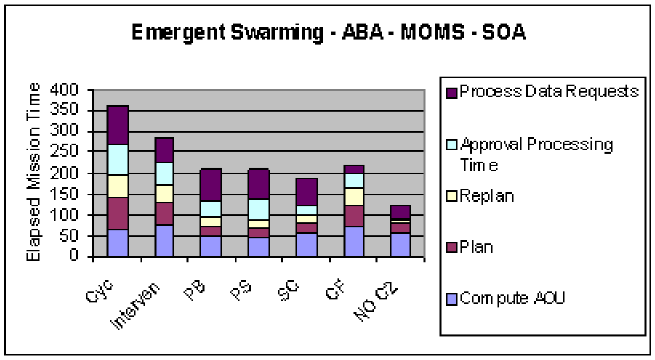
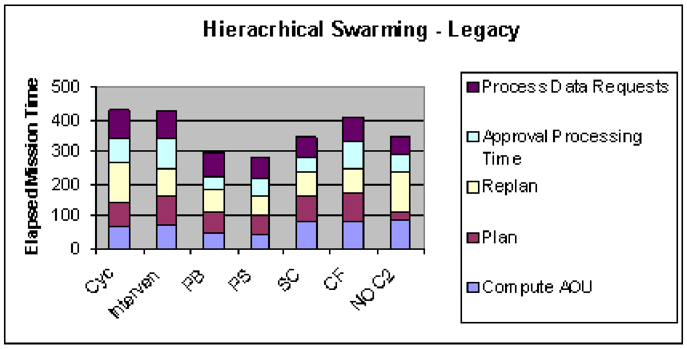


Figure 4 – Emergent Swarming Model Results

This model exhibited the fastest time to find the missing plane. However, it performed very inconsistently over multiple executions of the same mission. A hybrid of SOA, and Agent based models outperformed their legacy counterparts in all task categories. Since there was no leader, many duplicate plans and duplicate searches occurred increasing total mission time. However, the lack of a leader meant that no approvals were necessary which makes this one of the fastest models in terms of finding the plane if its first try, individualistic plans were accurate.

<u>Examining configuration 5 – Hierarchical Swarming</u>

The fifth configuration will be a hierarchical swarming model. Hierarchical Swarming[14] is closest to the traditional military C2 architectures, and this is because it represents an extremely good solution for dealing with complex problems. In Hierarchical Swarming, the nodes are organised into a hierarchy. In the event of nodes being lost, the hierarchy is maintained by promoting other nodes. Situational awareness information is fused going up the hierarchy, and at the same time, low-level tactical detail is dropped out. This means that the commanding node gets the "big picture" situation awareness that it needs. This simplifies the situational awareness fusion problem and avoids over-straining the information fusion capability of the nodes. The commanding node then produces a "big picture" plan (often called "intent"). This is passed down the hierarchy, and tactical detail is added by subordinate nodes. This avoids over-straining the planning capability of nodes. In the absence of computer technology, such a hierarchy has been the most effective mechanism of command. The leader drone after the start of mission creates a plan, determines the area of uncertainty for the search pattern, and requests the number of servant drones required, receives permission to use the drones, and issues tasking orders to each drone in terms of its own particular search pattern and assigns a protector drone.



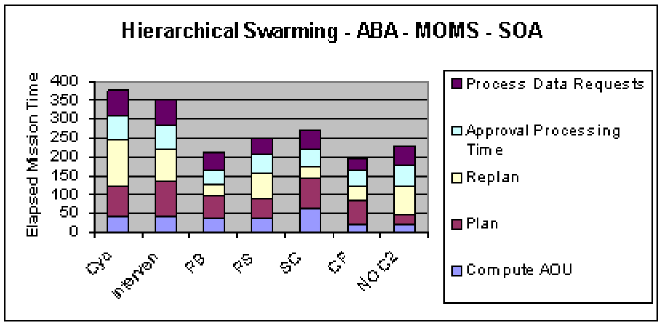


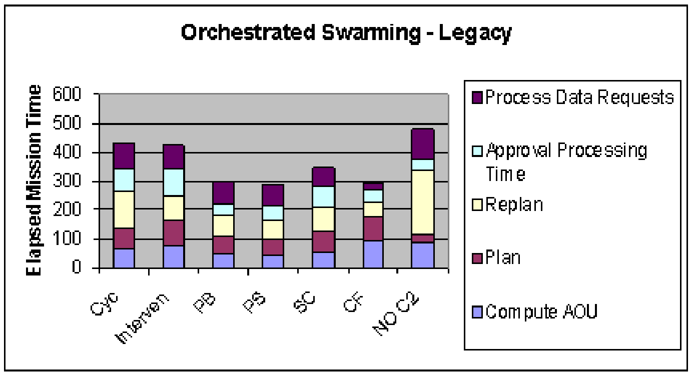
Figure 5 – Hierarchical Swarming Model Results

By analysis, the results for the hierarchical swarming configuration were that it outperformed other topologies in terms of consistently finding the missing plane during repeat identical runs. Results for the Hierarchical Swarm Operational Architecture using legacy indicate little value was maintained by varying the C2 structures. All C2 models improved their task times with the addition of the SOA for data request processing and the MOMS for approval time processing. Planning and re-planning also improved with the SOA since data was more readily accessible.

Control Free C2 with Hierarchical Swarming granted the most freedom to the closest tactical planners and required a minimum of C2 message requests while the SOA permitted decisive access to the knowledge stores of the hierarchy thus enabling particle swarm optimization. The MOMS architecture enables superior message traffic management and syndication while the agents were granted reach back to the knowledge stores of previous missions, contained in the hierarchies.

Examining configuration 6 – Orchestrated Swarming

The sixth configuration will be orchestrated swarming model. In Orchestrated Swarming[15], one of the nodes is chosen as a temporary "leader." In the Centralized Architecture, the C2 node was the node best equipped for command and control activities, but in swarming architectures, all the nodes are identical. The choice of "leader" is therefore made on the basis of suitable position, current combat situation, or other transient factors. This approach is sometimes used in Special Forces teams, where members can, if necessary, take over command from the nominal commander. Sensor data is sent to the "leader" node, where it is fused to produce an integrated situational awareness picture and an integrated plan of action. These are then broadcast to the other nodes. If the leader is unable to continue for any reason, the nodes agree on a replacement, which takes up where the previous leader left off. This approach limits network traffic, but it puts great stress on the C2 capability of the leader…. This option is therefore not suitable for very difficult problems, or for a very large number of nodes. However, Orchestrated Swarming potentially produces better plans than other Swarming techniques, provided that the C2 capability of the "leader" is not overwhelmed. The leader drone after start of mission creates a plan, determines the area of uncertainty for the search pattern, requests the number of servant drones required, receives permission to use the drones, and issues tasking orders to each drone in terms of its own particular search pattern. No protector is assigned. The summary results for this model are:



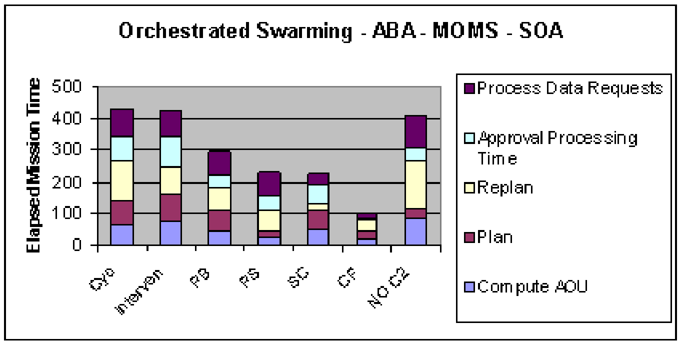
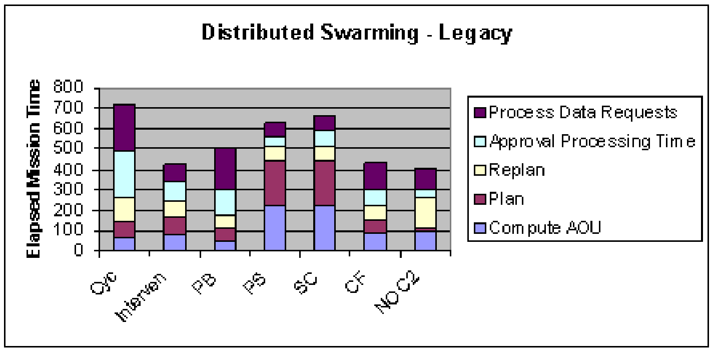


Figure 6 – Orchestrated Swarming Model Results

Results for the Orchestrated Swarm Operational Architecture using legacy indicate little value was maintained by varying the C2 structures. Please not the similarity in improvement by moving away from legacy systems to more modern software architectures which has been a consistent theme of this analysis. All C2 models improved their task times with the addition of the SOA for data request processing and the MOMS for approval time processing. Planning and re-planning also improved with the SOA since data was more readily accessible. Orchestrated found the plane, more consistently than any other architecture. We believe that this is due to the nature of the orchestrated swarm "voting" on its most competent leaders versus the second best solution which is the hierarchical approach. This is much more in line with particle swarm optimization than any other configurations except for leaderless. Also, the swarm leaders were assumed to be smart enough to select the best possible leader given the criteria of most suitable position and current combat situation.

Examining configuration 7 – Distributed Swarming

The seventh configuration will be a distributed swarming model. Distributed Swarming[16] has no "leader" role, and all decisions are made through consensus. Situational awareness is handled by all nodes broadcasting their sensor information, so that every node builds up an individual situational awareness picture. This generates a large amount of network traffic, but if the network can handle the traffic, it is extremely fast. There are two ways of handling planning with a distributed swarming architecture. The first has been called “collective” or “borg” decision-making (Wheeler & White 2004). In this style, each node goes through exactly the same decision process that the “leader” would have gone through if there was one, and then carries out the role that it assigns itself. This strategy only works for simple problems, where there is a single best decision, and each node therefore comes up with the same plan. This strategy also requires each node to have exactly the same situational awareness information, and to make decisions in a totally predictable way. For most military problems, these circumstances will not apply. Each drone must plan for itself. The summary results for this model are:



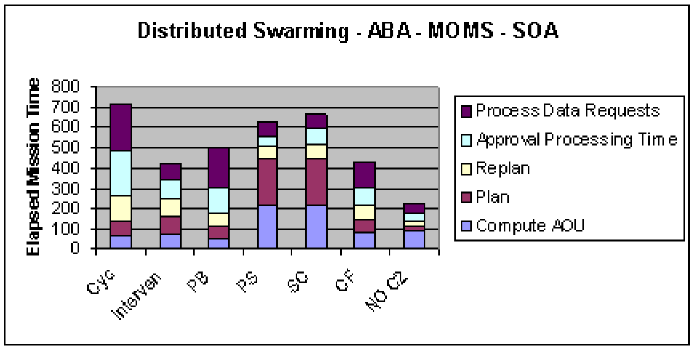


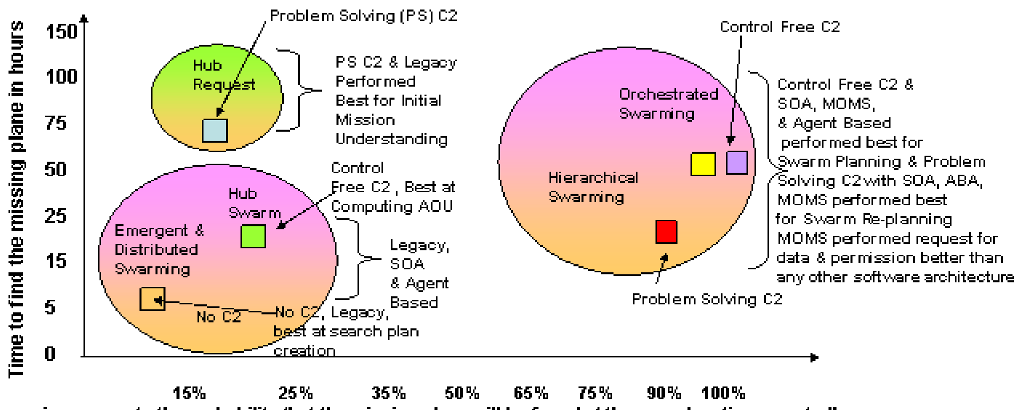
Figure 7 – Distributed Swarming Model Results

Results for the Distributed Swarm Operational Architecture primarily indicate the problem with a totally individualistic solution, namely that many hours of wasted search patterns occur. This forces the very large planning and re-planning times using legacy software. However, as with the other models, most task execution times were reduced by moving to the SOA, EBA, and MOMS hybrid approach.

Overall Results

The general results were surprising. At the individual task level, no single C2 paradigm stood out and no single software architecture paradigm stood out. Some C2 models are better than others given specific tasks and the same is true for software models in that they perform well in one task and poorly in another. The results of the analysis demonstrated that Dekker's orchestrated swarming model (consisting of all type 1 drones with one drone selected as a leader, and no protector drone), using a hybrid SOA, MOMS, and Agent Based Software Architecture, combined with a C2 model of control free structures performed best overall in repeatedly and consistently finding the missing plane in the required time. It must be pointed out that the agent based architectures excelled because they were capable of learning and adjusting plans, both individually and collectively. This seems to have been one of the most important advantages over emergent configurations in that the emergent swarms often chose the same failed plan during the re-planning task. The figure below depicts the collective best results from all the simulation runs and SME analysis. Emergent and distributed swarming models, with the extra search drone (no protector done required) found the missing plane in the shortest time, but not consistently. The Hub request model consisting of both smart and dumb drones also found the plane quickly but not consistently. The hierarchical swarm model found the missing plane approximately 75% of the time, with the orchestrated swarming model being the winner in consistently finding the missing plane on repeat simulation runs. Control free by far was the best command and control model for the varied operational configurations.

Solution Space of Most Reliable Configurations of Operational Architectures, Command and Control Models, and Software Architectures. The color coded tasks in each indicate the solution space region where optimal performance for that task was achieved given a particular C2, Dekker, and Software configuration



The x axis represents the probability that the missing plane will be found at the same location repeatedly.
Results:
Emergent & Distributed Swarming found the plane the fastest but least consistently. They also had the fastest time to process requests because there was no C2 in place eliminating approvals from a chain of command.
The hub swarm with control free C2 and Legacy, SOA, ABA, excelled at AOU computation. Hub request excelled at initial mission understanding with PS C2 and a Legacy Software Architecture - Hierarchical and Orchestrated Swarming found the plane most consistently using SOA, MOMS, & ABA software architectures. Orchestrated Swarming excelled at swarm planning and request processing while hierarchical swarming excelled at swarm re-planning using a PS C2 model. The hubs were also inconsistent because they had to assign a protector drone and thus had fewer search assets. Thus, in the end planning & C2 mattered in arriving at consistent repeatable results

Task Legend

Time to Understand Mission	🟦	Time to Compute Area of Uncertainty	🟩
Time to Create a Search Plan	🟧	Time to Process Requests	🟨
Time to Re-Plan	🟥	Swarm Planning Time	🟪

Figure 8 – Summary Results

Command and Control Model Results
Command and control in a minimum form (Control Free) seemed more successful than individuals with their own plans. The individualistic model resulted in many duplicated search patterns. The leaderless C2 models also resulted in less than optimal resource utilization resulting in more frequent re-planning and longer times to successfully complete a search.

Software Architecture Results
The Software Architectural analysis results were disappointing. Although one could intuitively anticipate some of the results, no single architectural model could achieve the best scores in all of the metrics classes. A hybrid of SOA, MOMS, and ABA performed optimally against single architectural paradigms.
The best software architectural configuration for computationally intensive tasks on a hub configuration is the legacy systems that were designed to embellish such processing.
The best software architectural configuration for initial planning was the agent based design which requires an SOA. The SOA permitted the agents a reach back capability that none of the other software architecture models could support. Federated query capability, for the discovery of useful knowledge needed for planning, was invaluable in reducing planning & re-planning time. The most surprising result was that the no single architecture infrastructure improved the results. What mattered was the design of the agents and their ability to communicate with each other independent if the underlying enabling agent infrastructure. When we dropped inter-agent communications, the agents performed worse than a traditional legacy system. Also disappointing was the lack of a standout metric performance for an EDA. I am offering the following as a hypothesis to explain this result. The time to process multiple database triggers and their related multiple stored procedures seemed to be the primary cause of the poor performance, slowing communications and consuming processing resources. However I must make it clear that from the limited metrics set we selected, much further research is required to form any definitive conclusions.
The SOA appears to have an advantage in reducing data collection times (thus knowledge accumulation time) but do not appear to offer any clear advantage in simple message communication tasks over the MOMS architecture. This makes sense given that agent based communication may be more peer to peer in nature where the act of communicating messages is the primary service required. A MOMS after all is optimized for this effort.

A few comments concerning capability portfolio management of assets and organizational competency

It is worth noting that the individual drone assets did not change in capability. This is an obvious but often overlooked aspect of NCW research. The configuration of the assets and how they were organized actually increased their collective capabilities. Orchestrated swarming can therefore be said to have exhibited an emergent capability of consistently finding the missing plane in time, this capability was not exhibited by the other configurations to the same degree. Yet all that changed was the organization and how they communicated, not the original capabilities of any single asset. It may be fair to state that individual competency and capability was increased by the re-organization of the assets and the methodology of permitting either more or less practical levels of individual freedom of action or more or less need for traditional chain of command approvals. This means that any capability portfolio analysis or competency assessments which do not take collective emergent behavior into account are at best going to cause budgetary overruns and at worst make procurement decisions to the detriment of the basic capability of the United States Military.

References

1. Dekker, Tony: "A Taxonomy of Network Centric Warfare Architectures", Defence Science & Technology Organization, DSTO Fern Hill , Department of Defence, Canberra ACT 2600, Australia, Email: tony.dekker@dsto.defence.gov. Cited with permission of the author, please read Mr. Dekker's full paper to receive a complete appreciation of his analysis, the paper may be found at:
2. A **thought experiment** (from the German term ***Gedankenexperiment***, coined by Hans Christian Ørsted) in the broadest sense is the use of an imagined scenario to help us understand the way things really are. Source: Wikipedia
3. The term swarm (schooling or swarming) is applied to fish, birds and insects and describes a behavior of an aggregation of animals of similar size and body orientation, generally cruising in the same direction. In Source:n.wikipedia.org/wiki/Swarm. In our case, we mean an aggregation of assets moving towards a solution point together. Swarm literally means a moving crowd. (JL)
4. **Particle swarm optimization** (PSO) is a form of swarm intelligence. Imagine a swarm of insects or a school of fish. If one sees a desirable path to go (e.g., for food, protection, etc.) the rest of the swarm will be able to follow quickly even if they are on the opposite side of the swarm. On the other hand, in order to facilitate felicitous exploration of the search space, typically one wants to have each particle to have a certain level of "craziness" or randomness in their movement, so that the movement of the swarm has a certain explorative capability: the swarm should be influenced by the rest of the swarm but also should independently explore to a certain extent. (This is a manifestation of the basic exploration-exploitation tradeoff that occurs in any search problem.) This is modeled by particles in multidimensional space that have a position and a velocity. These particles are flying through hyperspace (i.e., $\mathbb{R}^n$) and have two essential reasoning capabilities: their memory of their own best position and knowledge of the swarm's best, "best" simply meaning the position with the smallest objective value. Members of a swarm communicate good positions to each other and adjust their own position and velocity based on these good positions. There are two main ways this communication is done: a global best that is known to all and immediately updated when a new best position is found by any particle in the swarm "neighborhood" bests where each particle only immediately communicates with a subset of the swarm about best positions – Source: Wikipedia

5. Alberts, David, Hayes, Richard "Command Arrangements for Peace Operations", NDU Press, 1995
6. Lenahan, Jack, "Is the Service Oriented Architecture the only Valid approach to NCW", 2005 ICCRTS
7. Lenahan, Jack, "Assessing Self Organization and Emergence in C2 Systems and Processes", CCRTS June 2006
8. Lenahan, Jack, "An Abstract Process and Metrics Model for Evaluating Unified Command and Control - A Scenario and Technology Agnostic Approach", CCRTS, Jun 2005
9. Pacetti, Don, Lenahan, Jack, personal correspondence
10. Dekker, Tony: "A Taxonomy of Network Centric Warfare Architectures", Defence Science & Technology Organization, DSTO Fern Hill, Department of Defence, Canberra ACT 2600, Australia, page 11.
11. ibid
12. Dekker, Tony: "A Taxonomy of Network Centric Warfare Architectures", Defence Science & Technology Organization, DSTO Fern Hill, Department of Defence, Canberra ACT 2600, Australia, page 5.
13. Dekker, Tony: "A Taxonomy of Network Centric Warfare Architectures", Defence Science & Technology Organization, DSTO Fern Hill, Department of Defence, Canberra ACT 2600, Australia, page 11.
14. Dekker, Tony: "A Taxonomy of Network Centric Warfare Architectures", Defence Science & Technology Organization, DSTO Fern Hill, Department of Defence, Canberra ACT 2600, Australia, page 7.
15. Dekker, Tony: "A Taxonomy of Network Centric Warfare Architectures", Defence Science & Technology Organization, DSTO Fern Hill, Department of Defence, Canberra ACT 2600, Australia, page 8.
16. ibid

Glossary

The following information and definitions are provided as a short cut for the reader who may be unfamiliar with the work of Mr. Dekker or Dr. Alberts.

Dekker's Operational Architecture Taxonomy definitions from his paper[1] are as follows:

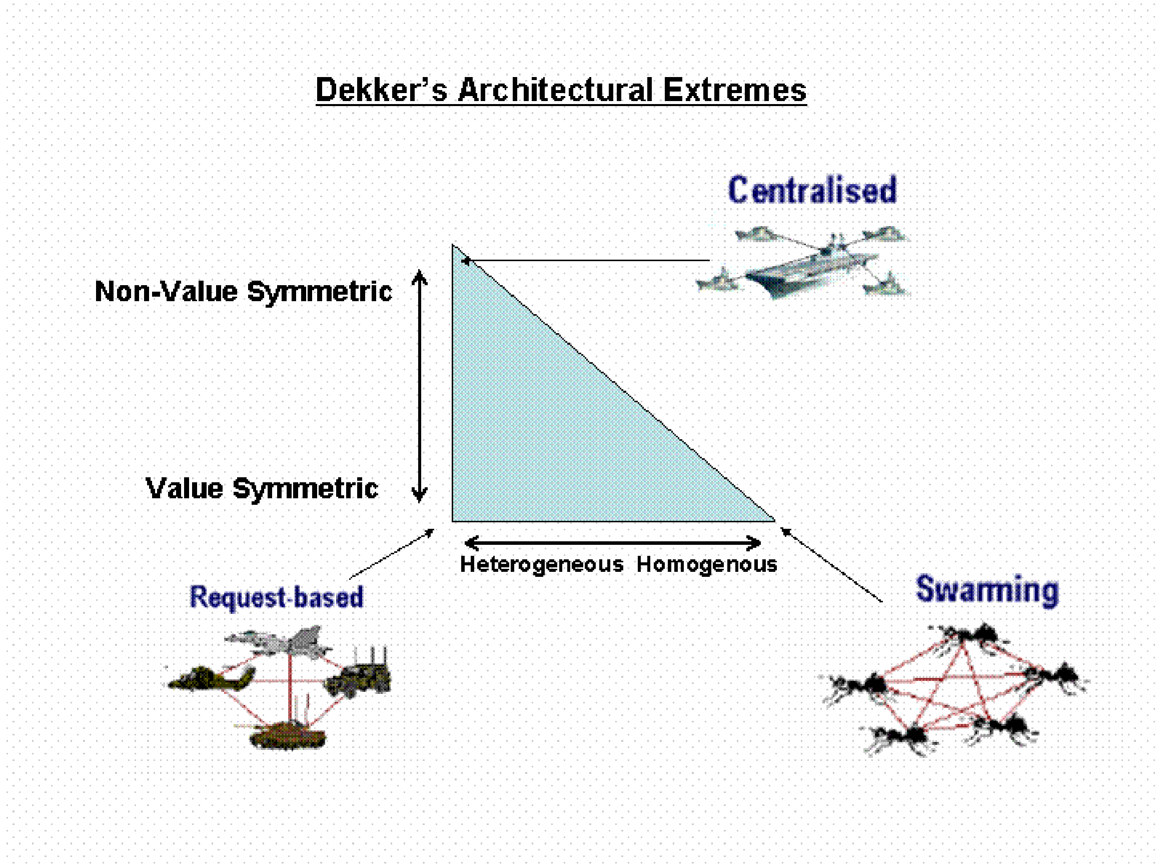


Figure 4 – Modified by this author from Dekker's original

Centralized - The most non-value-symmetric architecture (Type A) has a single high-value central "hub" node, surrounded by a cluster of nodes of lower value. If there are multiple high-value nodes, a Type B (Hub-Request) or Type D (Joint) architecture results. The central "hub" in a Type A architecture provides services of such high value that the force cannot operate effectively without it. The "hub" is therefore what Clausewitz called the "*centre of gravity... on which everything depends*" (Clausewitz Fully centralized C2 makes sense when the operational and tactical problems are suitable, when the "hub" has access to all the required information and has the necessary facilities for decision-making, and when the network allows centralized instructions to be disseminated sufficiently quickly (Dekker 2003*a*). Such fully centralized C2 appears to be more suited to the air and maritime environments than the land environment. Australia's commitment to Mission Command (Lind 1985, Australian Department of Defence 2002*b*), however, favors more decentralized

Hub Request - These architectures involve a mix of nodes of different kinds and values. Such a mix arises particularly in a Joint force, and involves mixing elements of all the other types of NCW. How this mixture should be achieved is dependent on understanding fully the other types. High-value nodes within a Joint architecture will behave more or less like "hubs." Groups of similar nodes will to some extent display Swarming behaviour. In addition, requests will be passed between different kinds of node. Achieving such a "seamless".

Request Based The combination of fully value-symmetric and heterogeneous forces is a collection of pure specialists, all different, but all of equal value. Each node does only a few things, and does them extremely well. Since military operations require multiple coordinated tasks, each node must call on many others to perform tasks that it cannot do. In this kind of architecture, requests for services are broadcast across the network, and the network identifies possible nodes which can satisfy the request (Hall *et al* 2004). These nodes in turn may require additional services.

Emergent Swarming Emergent Swarming occurs in nature among insects such as ants (Gordon 1999): "The basic mystery about ant colonies is that there is no management. A functioning organization with no one in charge is so unlike the way humans operate as to be virtually inconceivable. There is no central control. ... No ant is able to assess the global needs of the colony, or to count how many workers are engaged in each task and decide how many should be allocated differently. The capacities of individuals are limited. Each worker need make only fairly simple decisions." For example, in far northern Australia, "magnetic termites" build large termite mounds which are oriented north-south and contain a complex ventilation system which controls temperature, humidity, and oxygen levels. But termite brains are too small to store a plan for such a complex system, and since they are blind, they have no situational awareness of how much progress they are making. Instead, the termite mound structure emerges as a result of the termites following very simple rules, and exchanging very simple pheromone signals (Solé & Goodwin 2000). This style of operation has received considerable interest in the United States (US ASD C3I 2003). Although it may suit termites, it does not suit human beings (at least in Western armed forces). Following mindless rules without situational awareness would be tremendously corrosive of morale in a combat situation, and would have significant risks, such as that of reinforcing defeat. However, this style of operation is ideal for low-cost autonomous aerial (UAV), underwater (UUV), or terrestrial robotic devices. One possible example is the Area Dominance Munition (ADM), which the US Air Force is developing (Jane's 2003*a*). This is an expendable air-delivered UAV designed to loiter over enemy lines, and deploy multi-purpose shaped-charge warheads when targets are detected. Finding targets is done by sharing the limited information collected by onboard sensors (although this does raise ROE issues). Early experience with termite-like behaviour in robots is promising (Holland & Melhuish 1999), but much work is still required. The rules which nodes would follow in emergent swarming would probably need to be fine-tuned beforehand using, for example, genetic algorithms (Goldberg 1989, Smith *et al* 2004). This is a process which mimics evolution in nature and has proven itself very successful in making difficult design decisions. The evolution process would need to be combined with an agent-based simulation environment, in order to evaluate

Situational Aware Swarming
Orchestrated Swarming – In Orchestrated Swarming, one of the nodes is chosen as a temporary "leader." In the Centralized Architecture, the C2 node was the node best equipped for command and control activities, but in Swarming Architectures, all the nodes are identical. The choice of "leader" is therefore made on the basis of suitable position, current combat situation, or other transient factors. This approach is sometimes used in Special Forces teams, where members can, if necessary, take over command from the nominal commander. Sensor data is sent to the "leader" node, where it is fused to produce an integrated situational awareness picture and an integrated plan of action. These are then broadcast to the other nodes. If the leader is unable to continue for any reason, the nodes agree on a replacement, which takes up where the previous leader left off. This approach limits network traffic, but it puts great stress on the C2 capability of the leader.

Hierarchical Swarming — Hierarchical Swarming is closest to the traditional military C2 architectures, and this is because it represents an extremely good solution for dealing with complex problems. Of course, the people in a traditional military hierarchy are not identical "nodes," but something resembling Hierarchical Swarming is used because human beings share many of the same limitations. In Hierarchical Swarming, the nodes are organised into a hierarchy. In the event of nodes being lost, the hierarchy is maintained by promoting other nodes. Situational awareness information is fused going up the hierarchy, and at the same time, low-level tactical detail is dropped out. This means that the commanding node gets the "big picture" situation awareness that it needs. This simplifies the situational awareness fusion problem and avoids over-straining the information fusion capability of nodes. The commanding node then produces a "big picture" plan (often called"intent"). This is passed down the hierarchy, and tactical detail is added by subordinate nodes. This avoids over-straining the planning capability of nodes. In the absence of computer technology, such a hierarchy has been the most effective mechanism of command.
Distributed Swarming Distributed Swarming has no "leader" role, and all decisions are made through consensus. Situational awareness is handled by all nodes broadcasting their sensor information, so that every node builds up an individual situational awareness picture. This generates a large amount of network traffic, but if the network can handle the traffic, it is extremely fast. There are two ways of handling planning with a distributed swarming architecture. The first has been called "collective" or "borg" decision-making (Wheeler & White 2004). In this style, each node goes through exactly the same decision process that the "leader" would have gone through if there was one, and then carries out the role that it assigns itself. This strategy only works for simple problems, where there is a single best decision, and each node therefore comes up with the same plan. This strategy also requires each node to have exactly the same situational awareness information, and to make decisions in a totally predictable way. For most military problems, these circumstances will not apply. The second style of planning in Distributed Swarming has been called Mission Agreement (Lambert & Scholz 2005) or

Negotiation (Dekker 2002). In this style, each node has its own individual plan, and these plans are synchronized with each other through a negotiation process.

Command and Control Types from the Alberts Paper[2]

Cyclic - The greatest degree of centralization occurs, however, when the senior headquarters issues orders to all subordinates, but does so on the basis of a preset cycle time. The Chinese Army and the Soviet World War II forces adopted this approach because their communications structures could not provide continuous information to the central headquarters and because their subordinate organizations were culturally unable to display initiative in the absence of detailed directives. The US Air Force has followed the same approach since World War II, but for a very different reason - the complexity of air operations has meant the information required, coordination needed, and relative scarcity of the assets involved tend to drive the decision making up the chain of command. The USAF has chosen to invest in communications systems so they can issue orders at the numbered Air Force level. The 24-hour air tasking order is **cyclic**, however, in part because the amount of processing

Control Free command centers (the most distributed approach) seek to assign missions to their subordinates, who are then expected to employ all the assets available to them to accomplish the missions. This requires a military organization where the lower echelons are competent and trusted implicitly by the higher echelons. The system designed by the Germans for World War II is the case that fits most clearly in this category. The success of Germany's blitzkrieg was due not only to the superior weapons and mobility of German forces, but also to the capacity of their officers and non-commissioned officers to operate independently, even under trying conditions. (The fact that Hitler and the Nazi Party often interfered with this system is one major reason that it did not work effectively all the time.)

Problem Solving - For their parts, the US Army and Navy have, since World War II, tended to issue problem-solving directives in which missions and objectives are articulated for two levels of subordinates and substantial guidance about how the objectives are to be achieved is also included. Although this approach provides more detailed direction than the UK philosophy, considerable room remains for lower-level initiative and creativity in accomplishing the objectives. At the same time, however, the high-technology assets which US forces tend to employ often mean that subordinates are heavily dependent on senior commanders for key assets such as lift, intelligence, supplies, or precision munitions.

Problem Bounding UK doctrine can best be understood as problem-bounding. That is, the higher headquarters tend to compose their directives in terms of the objectives to be accomplished, but to couch them in very general terms. Hence, directives are more specific than mere mission assignments and some explicit boundaries (deadlines for achieving some objectives, guidance on risks that might be accepted or avoided, etc.) are articulated. British plans for an operation tend to be less detailed than those of Americans, often by a factor of three to one, reflecting this lack of detail.

Interventionist - The Cold War era Soviet system, for example, can best be described as interventionist, in that it relied heavily on central authority to issue directives, but also maintained very detailed information about the battle (requiring continuous and specific reports from subordinates two layers down) and attempted centralized control through detailed directives. The Soviets used exercises and training of front line units to ensure that they could execute a variety of quite standard maneuvers, from breakthrough assaults and river crossings for land forces to standardized attack patterns against US carrier battlegroups at sea. Senior headquarters specified the time and place for such preplanned operations and controlled them through the preplanning process.

Selective Control - The Israelis admired the philosophy of the German approach, but felt that it was perhaps too decentralized, particularly given their narrow margin for error in wars that threatened the extinction of their country. They have developed selective-control systems in which higher headquarters also issue mission-type orders and expect subordinates to take broad and deep initiatives. However, their higher headquarters follow the battle in detail and are prepared to intervene in the event of a major opportunity or major threat that the lower level command does not perceive or cannot manage. This approach requires great discipline on the part of the senior commanders, who have tactical-level information and considerable skill as tactical commanders, but only intervene when operational or strategic level issues emerge. In essence, the Israelis prefer rapid reaction on the battlefield but seek to maintain the capability for central intervention.

Software Architecture Models

Agent Based Architecture

Cougaar defines an Agent as a software entity which autonomously communicates with other software Agents to achieve domain-specific functionality. Multiple agents often collaborate as peers in a Peer-to-Peer (P2P) distributed network. The complexity of each agent can range from simple embedded sensors to a highly complex artificial intelligence application. Each agent must be able to learn, may contain their own genetic algorithms, learning classifier systems (LCS) or may contain their own neural network topologies.

Event Driven Architecture

An Event Driven Architecture is a software architecture which is centered on the state changes (events) of an automata or database. The EDA is characterized by the initiation of message traffic due to the occurrence of change of a database row or rows through insertion, deletion, or updating. In the case of automaton state changes, events are usually described as program subroutines completing, object oriented methods completing, or the initiation of a request for data by a user. Errors and keyboard activities are also considered events which can trigger message traffic. The occurrence of the event usually results in the transmission of data in the form of a message, the primary direction of message flow is considered as a “push” rather than a “pull”. (Author’s definition)

Legacy System - *Legacy* software can be characterized informally as *old software that is still performing a useful job for the community*. Legacy software systems are programs that are still well used by the community or have some potential inherent value but that were developed years ago using early versions of Fortran or other languages. Source http://www.sesp.cse.clrc.ac.uk/Publications/Legacy-Software/Legacy-Software/node2.html

Message-oriented middleware comprises a category of inter-application communication software that generally relies on asynchronous message-passing as opposed to a request/response metaphor. Most message-oriented middleware (MOM) depends on a message queue system, although some implementations rely on broadcast or on multicast messaging systems. Source wiki

SOA – Service Oriented Architecture – is a software architecture which is layered upon an existing enabling software infrastructure, which delivers content and services to consumers primarily via the use of Web Services. Service Oriented Designs permit the simplification of content delivery services which can be used as components or "building blocks" by C4ISR mission architects. (Author's definition)
From web – (Service Oriented Architecture) - A system for linking resources on demand. In an SOA, resources are made available to other participants in the network as independent services that are accessed in a standardized way. This provides for more flexible loose coupling of resources than in traditional systems architectures.
Source - http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html