C2 in the Joint Task Force (JTF) Enterprise

Jay Bayne
Echelon 4 Corporation
1045 W. Glen Oaks Lane, Ste 202
Mequon, WI 53092-3477
262.240.2956
jbayne@echelon4.com

Donald Diggs
OASD/NII
C2 Policy
Arlington, VA 22202-4302
703.607.0654
donald.diggs@osd.mil

Abstract

Command and control (C2) comprises those policies (rules) and mechanisms (services), implemented through human and synthetic means, required for the effective governance of an enterprise. An *enterprise* is an arbitrary unit of organization tasked with governance of a mission. We present an *enterprise command and control* (EC2) *framework* designed to provide DOD enterprises, exemplified by a Joint Task Force (JTF), with shared network-accessible C2 services. A JTF is DOD's designated unit of organization responsible for mission-specific joint operations. JTF EC2 includes a precise definition of enterprise, an associated *enterprise command structure* (ECS) and a specific set of *control processing services* (CPS). The proposed framework is consistent with the DOD's stated goal of migrating to network-centric (i.e., GIG-mediated) operations (NCO). In support of collaboration in jointly managed activities, agile JTF enterprises benefit from service-oriented capabilities along their vertical command (accountability) axes and along their horizontal production (logistics) axes, allowing them to support a wider range of NCO functions while simultaneously participating in and influencing behavior of multiple communities of interest (COIs).

Keywords: Joint Task Force, Enterprise Command and Control, C2, Network-Centric Operations

Note: An extensive collection of references for this and companion papers related to the Enterprise Command and Control (EC2) Framework is available at http://www.echelon4.com/references.htm. Within this paper, citations are of the form "[n]," where "n" is an index within that online reference list.

© 2006 All Rights Reserved

Table of Contents

1 Introduction 3
2 Joint Task Force (JTF) Model 5
3 JTF Enterprise Structural Model 7
4 JTF EC2 Model 8
4.1 Enterprise Command Structure (ECS) 9
4.2 Control Processing Services (CPS) Framework 12
4.3 CPS Feedback Control 15
4.3.1 JTF Situation Assessment Services 16
4.3.2 JTF Plan Generation Services 19
4.3.3 JTF Plan Execution Services 23
5 JTF Performance Considerations 26
5.1 JTF Performance Metrics 26
5.2 JTF Utility Measures 27
6 Conclusions 29

Table of Figures

Figure 1 - Cyclic C2 Process 4
Figure 2 - Joint Task Force Structure Model 6
Figure 3 - Federated Systems of JTF Systems 8
Figure 4 - Federated Operating Context 8
Figure 5 – JTF Enterprise C2 Services 9
Figure 6 - Enterprise Command Structure (ECS) Model 10
Figure 7 - Nested Command Structures 11
Figure 8 – Recursive JTF Structure Model 12
Figure 9 – Generalized Enterprise Control Loop 13
Figure 10 – JTF JOPES/CAP Processes 13
Figure 11 – CPS C2 Information Flow 14
Figure 12 – JTF CPS Information Flow 15
Figure 13 - CPS Closed-Loop Control (aka, Joint Op Cycle) 15
Figure 14 – JTF Situation Assessment Services 17
Figure 15 - Generalized Courses of Action 19
Figure 16 - JTF Plan Generation Services 20
Figure 17 - Notional Policy Structure 20
Figure 18 - Notional Asset Structure 21
Figure 19 - JTF Plan Execution Services 23
Figure 20 – Notional Command Structure 24
Figure 21 - Notional Tasking Order (TO) Structure 26
Figure 22 - JTF Performance Indices 27
Figure 23 - Example Time-Utility Functions 28
Figure 24 - JTF Performance 28

© 2006 All Rights Reserved

1 Introduction

In support of fielding network-centric C2 application services, we wish to formalize the meaning of the term "command and control" (C2) and the core processes it implies. We are especially concerned with enterprise[1] C2 policies and mechanisms that lead to effective, scalable, and interoperable (i.e., collaborative) *joint task forces* (JTF). Our objective is to introduce a scalable DOD *enterprise command and control* (EC2) [2] framework[3] that supports network-interconnected organizations participating in joint operations. We believe shared "service-oriented" EC2 systems and services, including accompanying intra- and inter-enterprise performance metrics, are required for effective joint operations. This framework formalizes architecture for component-based and reusable EC2 systems and services.

The term C2 is semantically rich and therefore its use can generate a significant amount of confusion depending on one's traditions and objectives. A few perspectives found in DOD literature [93, 101, 111, 118] include:

- An abstract concept (e.g., "commander's intent")
- A generalized and informal management framework
- A set of processes (e.g., information technology applications)
- A set of policies (e.g., rules of engagement)
- The ways a given commander chooses to operate in theater
- The means employed by a given commander in exercising his/her authority
- The ways a Military Service, through its officer training programs, expects its commanders to function
- The ways a Joint Force operates in order to harmonize and synchronize its component forces
- Whatever reasonable and prudent governance practices may be required to achieve specific objectives

A common thread among these definitions is to evade being too specific, avoiding specification in order to sidestep bureaucratic or political friction or for fear of limiting prerogatives of commanders, Services or coalition members. Flexibility through minimal specification is assumed necessary for effective management of complex and often chaotic situations. Furthermore, given that C2 is associated with the "art of management," it is by its nature a "fuzzy" form of control. Consequently, there are many who argue that C2 cannot or should not be formalized. Some object to the idea of formalization for fear of confronting traditional norms of enterprise governance, norms established over decades or centuries. Others believe the issue is fundamentally organizational – that there are no agreed upon definitions of what the term enterprise – the "object" of C2 – means or what it is, either in its dimensionality in space and time or the scope of effects and the associated costs of prosecuting its missions.

These are valid concerns, but they do not represent valid reasons for avoiding opportunities to improve the effectiveness of collaborative net-centric C2. There are several reasons for this. First, we wish to develop practical solutions for joint C2 that

---

[1] The term *enterprise* refers to an arbitrary unit of organization, commissioned with specific goals and objectives, a command element and requisite capabilities.
[2] [7] Bayne, J.S, Theory of Enterprise Command and Control, October 2005, to be published
[3] The term *framework* refers to a defined set of patterns, components or elements, or a skeleton of a solution, from which a specific system or solution may be derived (see, for example, http://en.wikipedia.org/wiki/Framework).

© 2006 All Rights Reserved

must necessarily rely on hard, not fuzzy, definitions. Second, traditional views and modes of operation, fostered in a less chaotic, less demanding and less collaborative era, may not properly apply to new realities. Third, technology's inexorable march continually offers new capabilities that, to be utilized effectively, require "paradigm shifts" in perspective, policy and action. Fourth, systems science is more competent today in engineering solutions to complex matters than at any time in history. Lastly, there are compelling reasons (e.g., terrorist and natural disaster response) to interconnect and interoperate complex dynamic socio-political systems in real time and in joint fashion that allow for synchronized distributed control. Interconnected semi-independent systems require improved forms of enterprise governance with more formal interfaces. Traditional forms of C2, especially those loosely defined, provide limited guidance for addressing these new realities.

A more formal JTF EC2 framework, based on a unified theory of C2, predicts a number of benefits. First, implementations based on a common framework create an environment whereby applications (software-based C2 services) designed for one level of command can in principle be deployed at levels above and below it in the command hierarchy, providing common and reusable applications and services and benefit from scale economics. Second, an operator or commander trained to use EC2 services a one level can move among regional and functional JTFs, or ascend (or descend) to the next level of command and continue to be at home with the concepts, policies, mechanisms and operations. Third, EC2 systems whose implementations are valid (validated and verified) at a given level of command are technically valid at other levels. Fourth, documentation and training prepared for a given level, region or function are applicable at other levels, regions or functions with minor (syntactic) adjustments. Fifth, emergence of a common C2 lexicon (semantics) and set of operational C2 doctrines promotes cooperation and sharing and the development of ontologies related to governance. Moreover, an established framework allows (encourages) innovation and contributions by an expanding community of enterprises, both within the US Department of Defense and among its allies and private-sector suppliers.

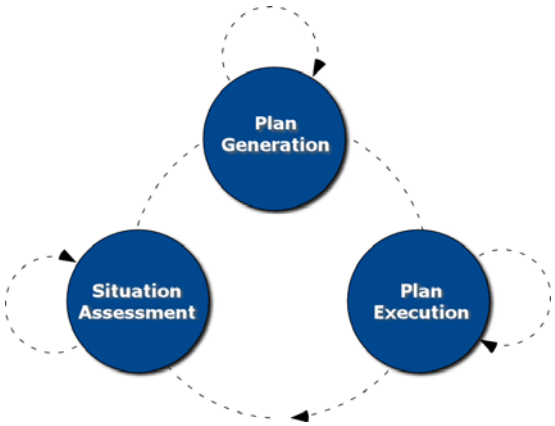

Figure 1 - Cyclic C2 Process

Simply defined, C2 comprises those policies and mechanisms required to effectively govern an enterprise. C2 is a cyclic process, depicted graphically in Figure 1, comprising three formal and iterative activities: situation assessment, plan generation and plan execution. Each of these activities require improved support from C2 services, some formal and supported by technology, some varying by enterprise and by mission, some automated and requiring minimal human

© 2006 All Rights Reserved

intervention and some ad hoc and largely social. Required improvements derive from increasing complexity of enterprise missions, improved accountability, increasing requirements for speed and agility, the need to collaborate with other governmental and non-governmental agencies - both domestic and foreign - and the quickening pace of situation awareness, decision and control. All benefit from higher greater degrees of C2 automation to assist the enterprise's human actors.

The definition and development of net-centric collaborative (i.e., interoperable) services-oriented C2 demand a higher degree of standardization than exists within today's closed, typically domain-specific C2 systems. Such standards in turn demand a more comprehensive framework for command and control than is provided by the traditional, social and *ad hoc* models of the past. Without higher and more robust forms of enterprise automation and control, the DOD will be progressively constrained in its individual and collective actions in an increasingly complex, interdependent and interactive world. An EC2 framework and corresponding technical implementation moves the frontier of enterprise command and control and its design and operation to a framework predicated on *open and shared* applications and services rather than *closed and isolated* systems.

Further, the proposed EC2 framework augments current DOD IT strategies [93, 95, 96, 111] by adding *C2 processes* to its current *communications*- and *information (data)-centric* orientation. Today, the DOD emphasizes data centricity as the cornerstone of its integration efforts. The ability to connect to and access arbitrarily large numbers of distributed and diverse information sources is a necessary and critical enabler of effective enterprise command and control. Large quantities of volatile context-sensitive information streaming in from various geographically distributed sources feed the "front-end" C2 processes of data fusion, sensory perception and situation assessment. Data centricity involves collateral issues of information pedigree, timeliness, accuracy, precision, semantics and a host of other matters that connectivity alone cannot resolve. These matters belong to the "down-stream" *EC2 processes* that ultimately assimilate, interpret and utilize the data.

In short, communications networks and their ability to support an abundance of data publishers and subscribers have a critical but not absolute role in enabling effective and collaborative enterprise C2. The EC2 framework, as applied to the JTF, motivates the next developmental phases of governance over the C2 domain and emphasizes the need to include the *assimilation* and effective *utilization* of information at the JTF. This allows for distributed command, institutional and collective awareness and the essential *processes* of C2 across the DOD enterprise, with specific emphasis on JTF concepts and their scaled deployment.

2 Joint Task Force (JTF) Model

Figure 2 diagrams the Joint Task Force (JTF) command structure as established by the Joint Chiefs of Staff (JCS)[4] [105, 106, 107]. The JTF relates two interdependent command authorities, an "establishing authority," the *superior* command and an "established authority," the *subordinate* command. The establishing authority may be the Joint Chiefs of Staff (JCS), Secretary of Defense (SecDef), the commander of a combatant command (COCOM) or subordinate unified command or an existing JTF. An established JTF may be temporary, existing only until achievement of specific goals or some situational end state, or it may be semi-permanent. Its duration is

[4] [105] DOD Joint Staff, "Joint Task Force Planning Guidance and Procedures," Joint Publication 5-00.2, Jan 13, 1999

© 2006 All Rights Reserved

typically a function of its mission and the evolving context in which it must operate. In general, a JTF is an embedded (subordinated) component of a larger establishment, ultimately governed by the President of the United States (POTUS) acting as Commander-in-Chief.

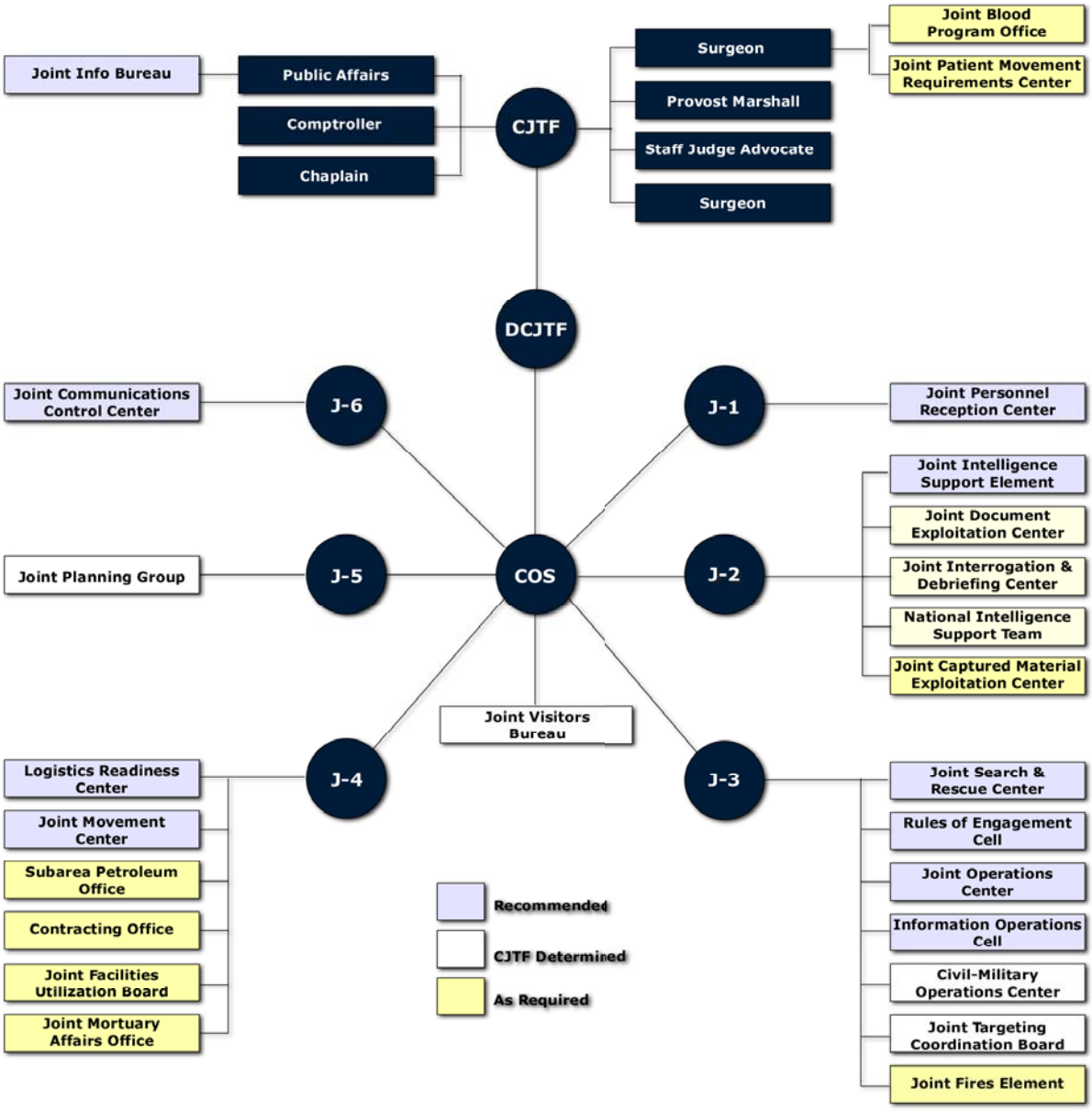

Figure 2 - Joint Task Force Structure Model

As diagrammed in Figure 2, when properly formed, the JTF command authority includes three principal actors governing J-coded staff directorates and supported by a number of additional functions, as summarized in Table 1.

© 2006 All Rights Reserved

JTF Staff Actors	JTF Staff Functions
CJTF	Commander
DCJTF	Deputy Commander
COS	Chief of Staff
J-1	Manpower & Personnel Directorate
J-2	Intelligence Directorate
J-3	Operations Directorate
J-4	Logistics Directorate
J-5	Planning Directorate
J-6	C4 Systems Directorate

Table 1 - JTF Staff Components

3 JTF Enterprise Structural Model

A JTF is an instrument of national policy, an enterprise designed for and capable of military force projection. As such, a JTF functions as a *unit of value production* [8, 9, 10], guided by and enforcing national policy objectives. An effective JTF 1) requires a well-defined mission aimed at achieving specified end-state goals (i.e., realizing specific *value propositions*), 2) must acquire and maintain sufficient capabilities (*assets*) and 3) is required to operate with agility and efficiency under evolving conditions and established *rules of engagement*.

As defined in JP 5-00.2 [107], JTF are large-scale (i.e., "heavy weight") units of organization, primarily addressing requirements of combatant commanders (2/3 star rank) dealing with broad objectives, generally within specific regional areas of responsibility (AOR). Additionally, a given JTF may perform as a *functional entity*, as in the USSTRATCOM Joint Functional Component Commander (JFCC) structure[5]. JP 5-00.2 states that a JTF may spawn subordinate JTF, but it offers no specific formula for coordinating JTF chains either in the vertical superior-subordinate (*command axis*) dimension or in the horizontal supplier-consumer[6] (*production axis*) dimensions. We introduce these two dimensions in the following section.

3.1 JTF Operational Context

The Joint Task Force is the organizational entity responsible for management of *joint* US warfighting activities, inherently collaborative affairs. Furthermore, JTF entities may operate in allied or coalition structures, as diagrammed in Figure 3. Here a single JTF belongs to three distinct COI, labeled A, B and C, performing at different levels of command in each. Such relationships constitute a *federated enterprise* where individual and mutual agendas collectively define the JTF mission.

A JTF is therefore an enterprise nested within another enterprise. As diagrammed in Figure 4, JTFs are members of one or more governmental or non-governmental organizations, federations, domains or communities of interest. The central enterprise in Figure 4 is labeled $JTF_{j,k,l}$. The index "j" identifies the operational domain, federation or community of interest (COI), "k" denotes the horizontal position in the federation's effects production network and "l" represents the location in its vertical command network. Within a given COI, the central JTF has a single *superior* designated as $JTF_{j,k,l+1}$ and potentially many *suppliers* designated as $JTF_{j,k-1,l}$,

[5] http://www.defenselink.mil/DODgc/olc/docs/test05-04-07Cartwright.doc
[6] The commercial supplier-consumer dimension, in military terms, is referred to as logistics, the flow along supply chains to/from a military objective (e.g., enemy, maneuver).

© 2006 All Rights Reserved

targets (commercial *clients* or military *adversaries*) designated as $JTF_{j,k+1,l}$ and *subordinates* designated as $JTF_{j,k,l-1}$. Notice in the figure that each supplier, client (target) and subordinate JTF may actually represent (i.e., be a proxy for) multiple peer-level JTFs.

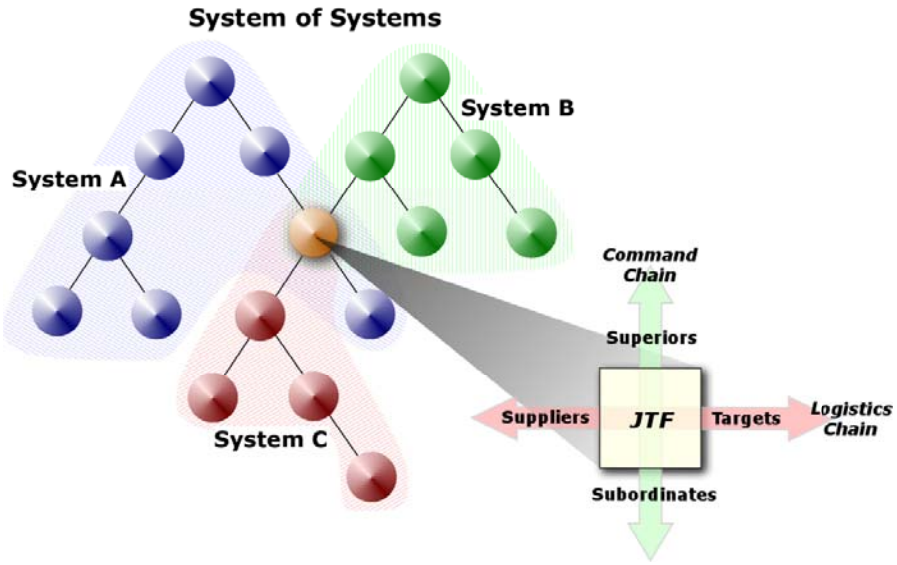

Figure 3 - Federated Systems of JTF Systems

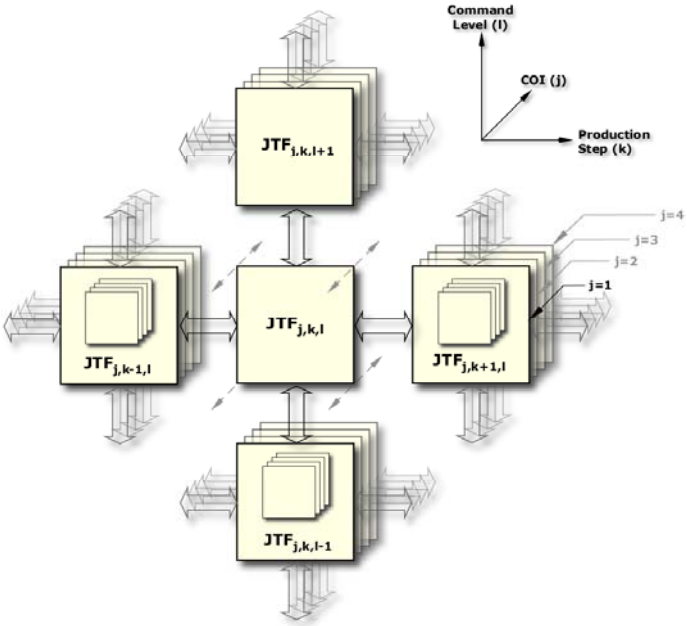

Figure 4 - Federated Operating Context

4 JTF EC2 Model

For a given mission, the operational behavior and effectiveness of JTF management are central to the success in achieving its desired end state objectives. Operational behavior is defined in terms of a core set of "control processing services" (CPS) [7,

© 2006 All Rights Reserved

8]. Its corresponding management organization is defined in terms of an "enterprise command structure" (ECS) [8, 9]. For JTFs to scale vertically up and down the command axis and horizontally along its production axis, CPS and ECS services must also scale.

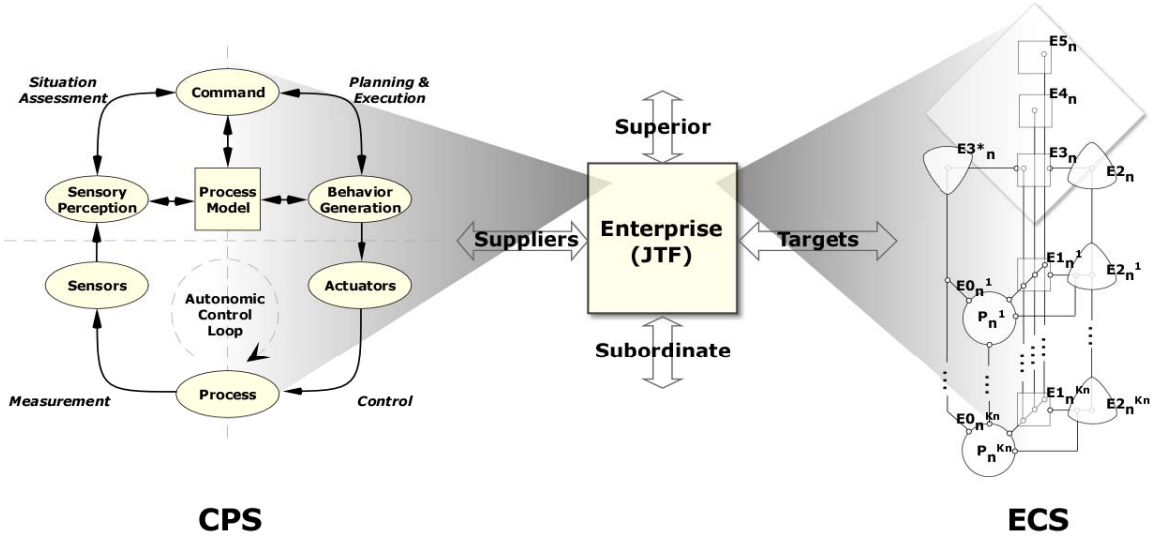

Figure 5 – JTF Enterprise C2 Services

Scalability, especially of *service-oriented software*[7] that provides for collaboration and coordination among distributed JTF, requires that the organization and operation of C2 systems exhibit a high degree of commonality and reusability. As presented in Figure 5, this requires that each JTF include complementary sets of Enterprise Command Structure (ECS) and a set of Control Processing Services (CPS) functionality[8].

4.1 Enterprise Command Structure (ECS)

The Enterprise Command Structure (ECS) establishes and formalizes the operational relationships between and among actors in JTF command enclaves, the people and supporting automated processes that are directly accountable for situation awareness, planning and execution of mission orders. Furthermore, ECS services must provide for a high degree of scalability, up and down the superior-subordinate command axis of an enterprise, as well as along its logistic (i.e., production) axis.

Details of the ECS model appear in several published papers [9, 10, 11], in R&D work performed under and documented within AFRL contracts F30602-03-C-0154 and FA8750-04-C-0084 [16, 17] and in the pre-publication version of the book *Theory of Enterprise Command and Control* [7]. Figure 6 and Table 2 summarize the ECS model.

[7] The EC2 model presumes a network-centric model of software enabled services. Such software is characterized as "service oriented" and derived from a "service oriented architecture" (SOA) based on specific internet protocols (e.g., SOAP, UDDI, XML, etc).
[8] This inclusion may be through separate implementations of common software modules or through the sharing of web-based SOA services.

© 2006 All Rights Reserved

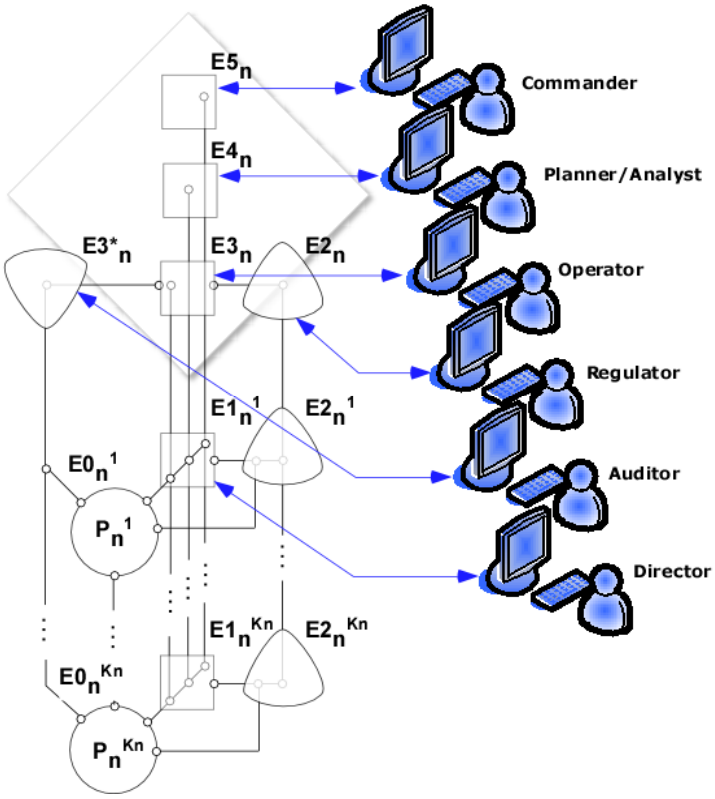

Figure 6 - Enterprise Command Structure (ECS) Model

Label	Services	Enterprise Roles & Responsibilities
E5	Commander	Mission Goals & Objectives, Authorization & CONOPs
E4	Analysis/Planning	Modeling, Situation Assessment & Plan Generation
E3	Operations	Plan Execution & Capability Management
E3*	Audit	Program & Process Performance Assessment
E2	Regulation	Plan (Task) & Resource Synchronization
E1	Direction	Plan (Task) Management
E0/P	Production Process	Embedded [Value] Production Process

Note 1: "*" designates a supporting role at a given echelon
Note 2: the subscript "n" in Figure 6 denotes command level
Note 3: the superscript "K_n" in Figure 2 denotes the number of embedded JTFs

Table 2 - Principle ECS Command Actors

Salient features of the ECS model include:

1. Its self-similar structure, the principal source of its scalability; within Figure 6 the command level comprises the E5-E4-E3 group, a group identified by the enclosing rectangle at the top of the figure. Subordinate (i.e., embedded) commands are identified by the group E1-E0. These subordinate structures recurse, like fractals, and are structurally equivalent to their containers.
2. Its ability to synchronize (ref. E2 and E3* actors), in both time and with respect to shared serially reusable resources, multiple subordinate and allied peer-level enterprises, and

© 2006 All Rights Reserved

3. Its ability to monitor (audit) the performance of its concurrent distributed operations using metrics that are shared among federation members.[9]

Principal actors in the JTF model align with their ECS model counterparts as described in Table 3.

ECS Element	JTF Command Element	Command Function
E5	CJTF	Commander
E4	DCJTF + J5 JPG	Deputy Commander
E3	COS + J3	Chief of Staff
E3*	JOC	Joint Operations Center
E2	JFE	Joint Fires Element
E1-E0	JFACC	Joint Force Air Component Command
E1-E0	JFLCC	Joint Force Land Component Command
E1-E0	JFMCC	Joint Force Maritime Component Command
E1-E0	JSOTF	Joint Special Operations Task Force
E1-E0	JPOTF	Joint Psyops Task Force

Table 3 - JTF-ECS Relationship

A key feature of the ECS model is its symmetry; a feature that allows it to be applied recursively (ref. Figure 7) to an enterprise's operating structure. The figure depicts four levels of successively (left to right) subordinated command, beginning at level "n" and ending at level "n-3.

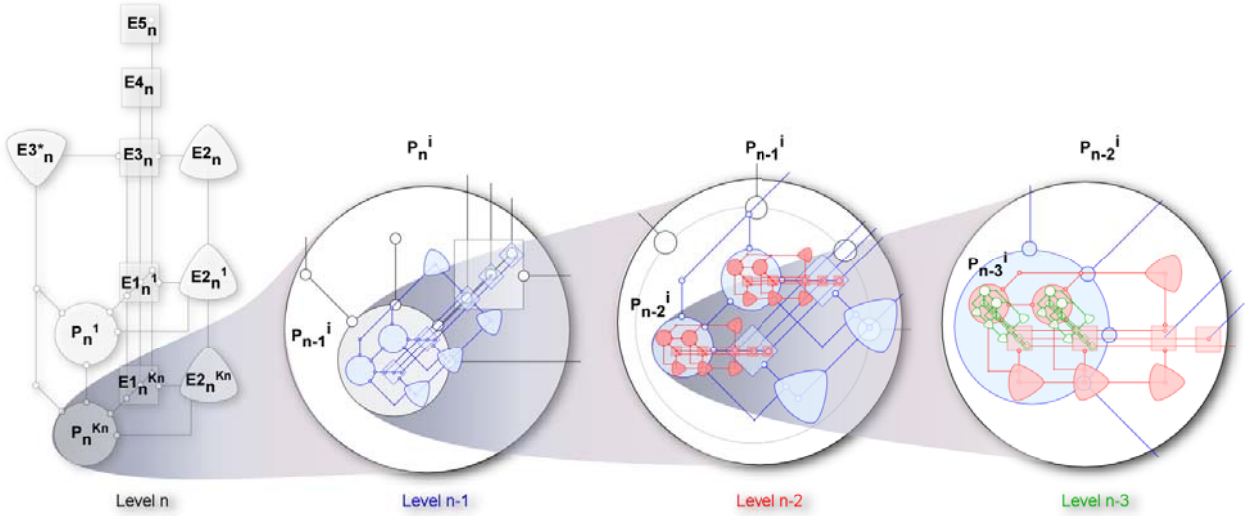

Figure 7 - Nested Command Structures

Note that such a recursively self-similar command structure exhibits collateral benefits. Commanders at a given level will know and better anticipate the structure and operational requirements of their superiors and subordinates. Training and corresponding documentation requirements are common, therefore reusable and more widely applicable. Software upgrades at one level of command are applicable at adjacent levels of command, all lowering the per-JTF costs basis. Policies and procedures scale. Common lexicon used in intra- and inter-command

[9] [7, 8] introduces an integrated set of *performance measurement services* (PMS) that implement the theory's scale free measurement system

© 2006 All Rights Reserved

communications (e.g., tasking orders) will emerge, facilitating development of ontologies, reusable lessons-learned, and coherent historical records.

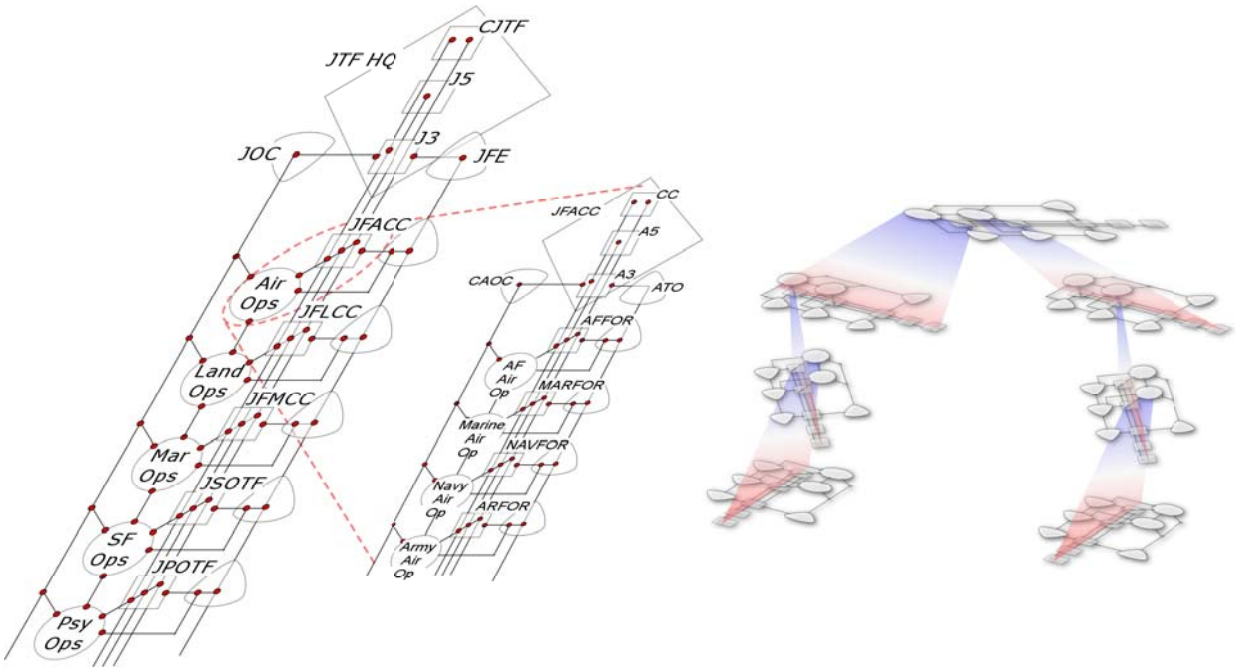

Figure 8 – Recursive JTF Structure Model

With reference to the JTF component model shown in Figure 2, Figure 8 depicts the nested structure of a JTF command, including its embedded Joint Force Air Component Command (JFACC) using the ECS modeling concept.

4.2 Control Processing Services (CPS) Framework

JP 5-00.2 discusses JTF control processing requirements that have evolved along traditional lines of military operations, supported by J-6 C4 systems and software services (e.g., GCCS/JOPES). The basic function of an enterprise, once commissioned with orders and resources, is too continuously (iteratively) perform situation assessment, plan generation and plan execution. This sequence defines a generalized "enterprise control loop," introduced in Figure 1 and drawn with detail in Figure 9. Both figures represent core C2 activities found in traditional models[10].

The JTF JP 5-00.2 specification defines its Joint Operations, Planning and Execution System (JOPES) control loop services, in relation to its Crisis Action Planning (CAP) processes, as depicted in Figure 10.

[10] Boyd's OODA Loop, Wohl's SHOR Model, Lawson's C2 Process Model and the JOPES services presently deployed in the DOD's Global Command and Control System (GCCS)

© 2006 All Rights Reserved

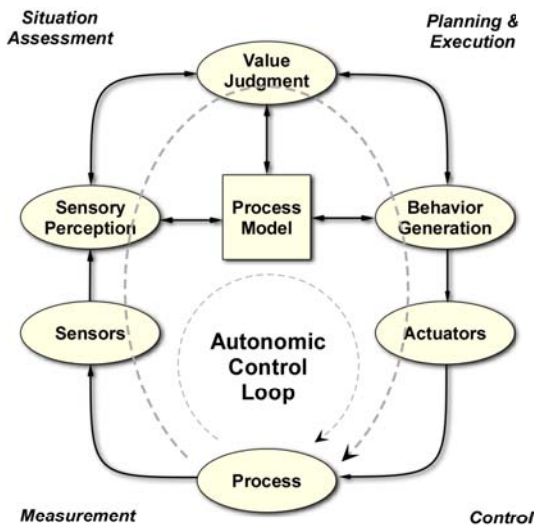

Figure 9 – Generalized Enterprise Control Loop

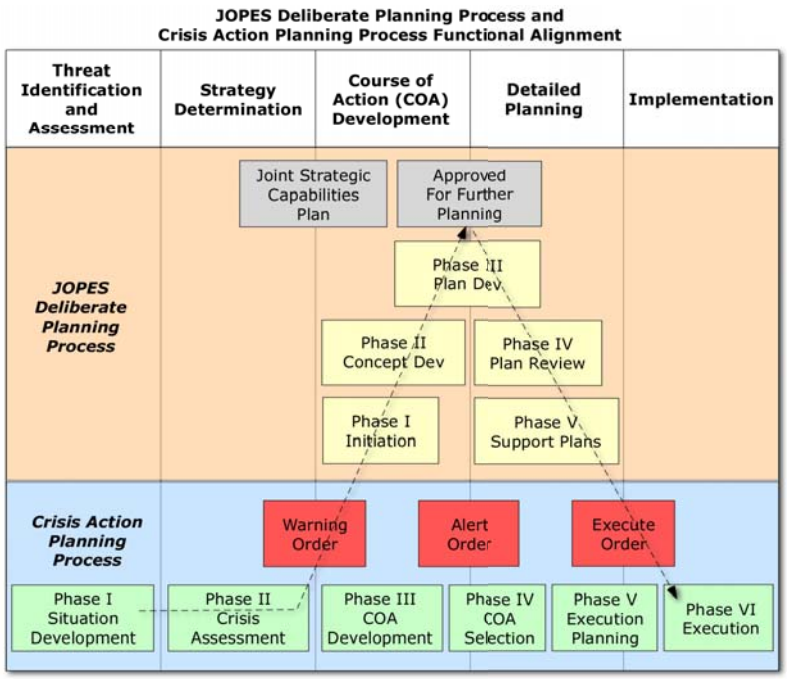

Figure 10 – JTF JOPES/CAP Processes

By comparison, the CPS model defines its enterprise C2 processes and associated information flows, in relation to the standard C2 loop, in Figure 11. The control loop comprises seven stages organized in three service groups: situation assessment services (SAS), plan generation services (PGS), and plan execution services (PES). Table 4 summarizes these services.

Table 4 also defines the inter-stage information flows. A key element of the model, however, is that each stage may act as both a subscriber to and a publisher of these flows. This allows each JTF to participate in its own ECS activities, but also those of its federation allies. This aspect of the CPS model represents one of several

© 2006 All Rights Reserved

important extensions to the JOPES model defined in the JP 5-00.2 specification. Figure 12 compares the EC2 and JOPES control processing services.

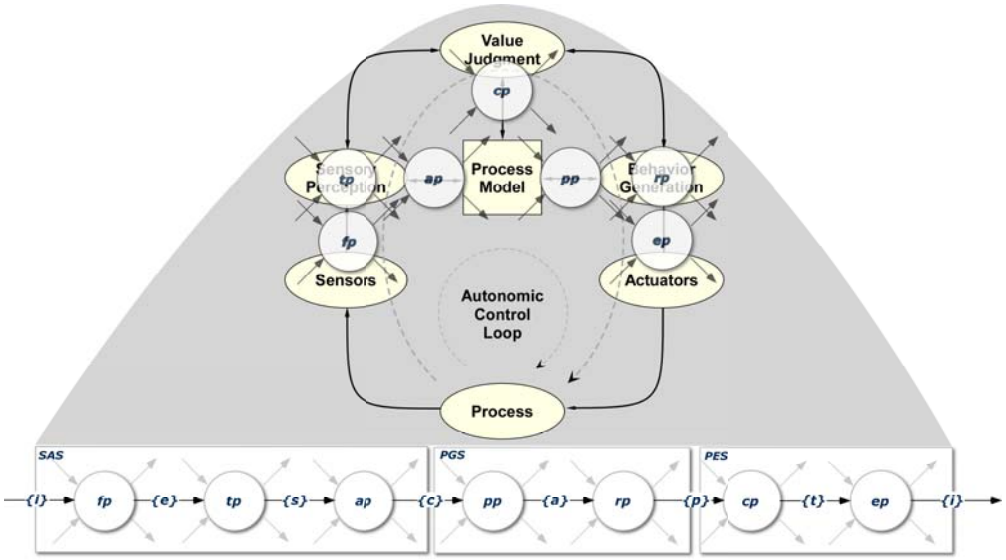

Figure 11 – CPS C2 Information Flow

	CPF Stage	Stage Services	Stage Inputs	Stage Outputs
SAS {i} fp {e}	**Filter Process**	Processing information feeds {i} from sources in order to identify information objects of interest that produce subsequent *alarm and event notifications {e}*	Information sources {i}, including RSS feeds, web services, databases, and related voice, video and data sources.	Alarm and event notifications objects {e}
{e} tp {s}	**Triage Process**	Processing sequences of alarm and event notifications {e} in order to identify *situations* {s} related to the VPU's current tasks	Alarm and event notification objects {e}	Situation notification objects {s}
{s} ap {c}	**Analysis Process**	Processing identified situations {s} and associating them with one or more potential *courses of action* (COA) {c} with which to respond	Situation notification objects {s}	COA notification objects {c}
PGS {c} pp {a}	**Policy Process**	Identifying all policies that govern COA {c} in order to evaluate compliance and risk, reducing the list of possible COA to single *plan of action* (POA) {a}	COA notification objects {c}	POA notification objects {a}
{a} rp {p}	**Resource Process**	Identifying resources required of POA {a} in order to properly "fund" it, converting it to an executable *plan of record* (POR) {p}	POA notification objects {a}	POR notification objects {p}
PES {p} cp {t}	**Command Process**	Authorization of new POR {p} such that they "fit" into current activities by developing a new schedules and issuing new tasking orders (TO) {t}	POR notification objects {p}	TO notification objects {t}
{t} ep {i}	**Control (Execution) Process**	Execution of authorized TO {t}, including management and synchronization of shared resources, producing effects-based results {i}	TO notification objects {t}	Information notification objects {i}

Table 4 - Control Processing Services (CPS)

© 2006 All Rights Reserved

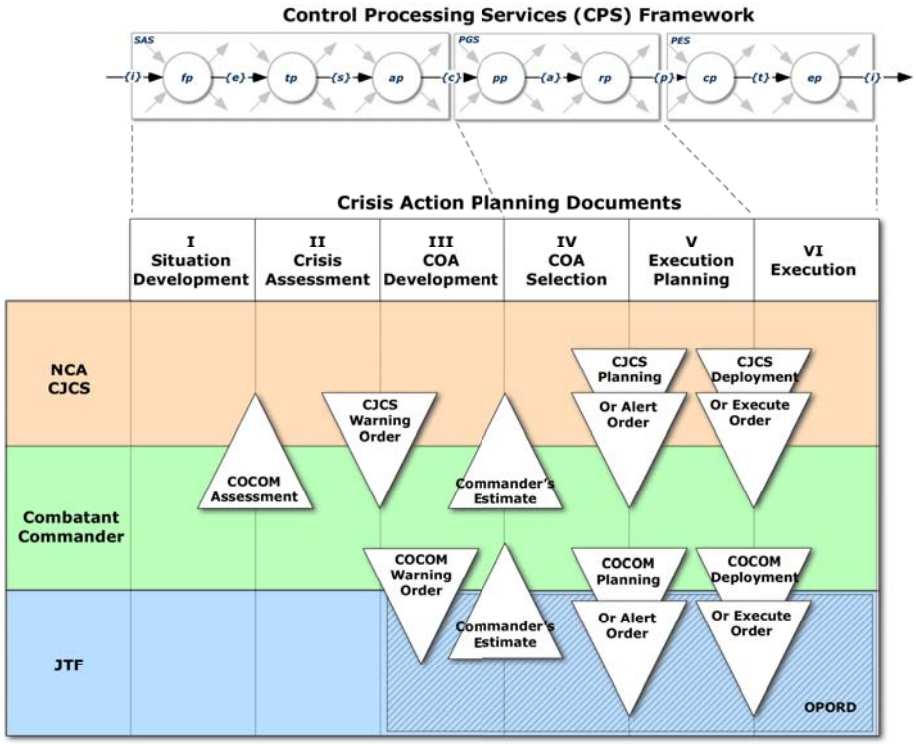

Figure 12 – JTF CPS Information Flow

4.3 CPS Feedback Control

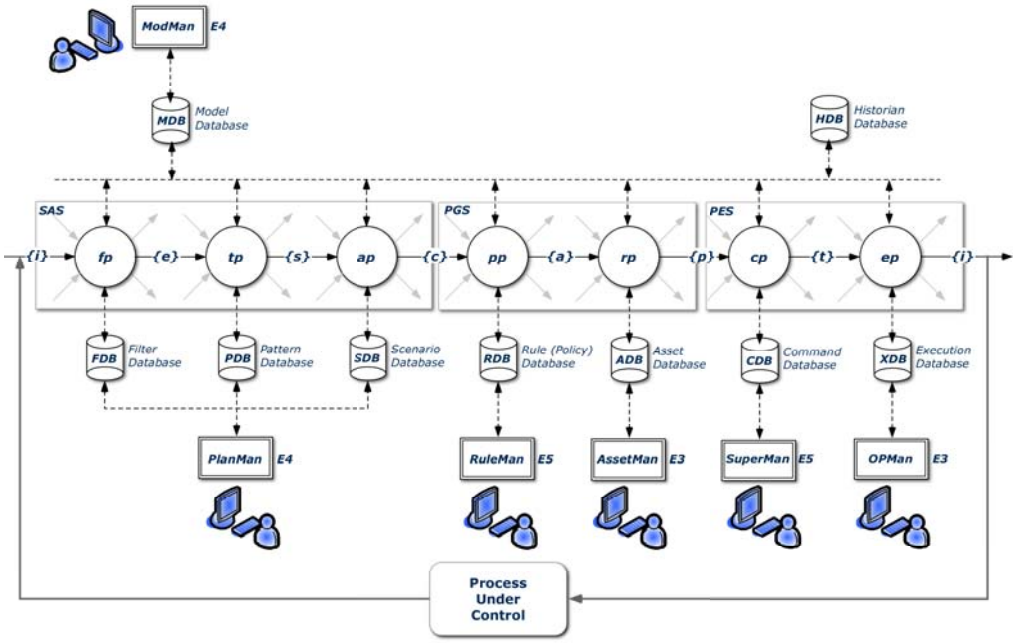

Figure 13 - CPS Closed-Loop Control (aka, Joint Op Cycle)

© 2006 All Rights Reserved

Among its primary features, CPS provides feedback to the command enclave, as shown in Figure 13. The JTF must provide the same capability, but its specification is silent on how, via what channels and protocols, or with what performance requirements such feedback emerges. The CPS model is explicit, at least as to the flow of information into the enclave and its processing. The loop closes at a rate commensurate with the processes under its control. JP 5-00.2 describes a "daily op cycle," the minimum default cyclic processing within the JTF.

4.3.1 JTF Situation Assessment Services

With reference to Figure 14, the end-to-end transformation for the SAS stage is:

$$\{c\} = respond[\{assess[\{detect[\{\{i\}, FDB]\}, XDB]\}, SDB]$$

{c} is a list of possible courses of action, *{i}* is an input information list, FDB is the filter database, XDB is an event pattern database and SDB is a scenario (COA) database.

Situation assessment services provide the processes of

1) Subscribing and
2) Listening to relevant information sources,
3) Identifying in the streams from those sources relevant events of interest,
4) Correlating sequences of events within and across streams in order to identify the occurrence of a new situation or the change in state of an existing situation, and
5) Identifying one or more potential reactions (courses of action) to these situations

These services provide institutional *sensory perception*, the critical front-end processing for downstream C2 services of planning and plan execution.

4.3.1.1 Filtering and Subscription Services

Effective enterprise C2 relies on information about the external world and about the status of internal enterprise capabilities. Acquisition of knowledge about the external world is achieved through *intelligence, surveillance and reconnaissance* (ISR). In a net-centric environment, ISR and other Joint Planning Group (JPG) functions *publish*, with various security restrictions, information about the processes (situations) of interest to one or more enterprises. The number and voracity of enterprises that demand such knowledge define a "market" for information. We do not discuss this market place or its dynamics. However, we assume such markets exist and that they provide their information products through publications.

Publications are typically organized according to *topics*. Enterprises subscribe to topics of interest, with access governed by their respective *access rights*. Subscriptions (streams) enter a JTF through the filter process (fp) interfaces. Publications exit a JTF at interfaces at each stage. Methods for publishing, discovery, topic registration and subscribing and access control are well known within the GIG computing and communications community. They are not discussed here, but referred to collectively as issues related to *information assurance* and *publish-subscribe protocols*.

© 2006 All Rights Reserved

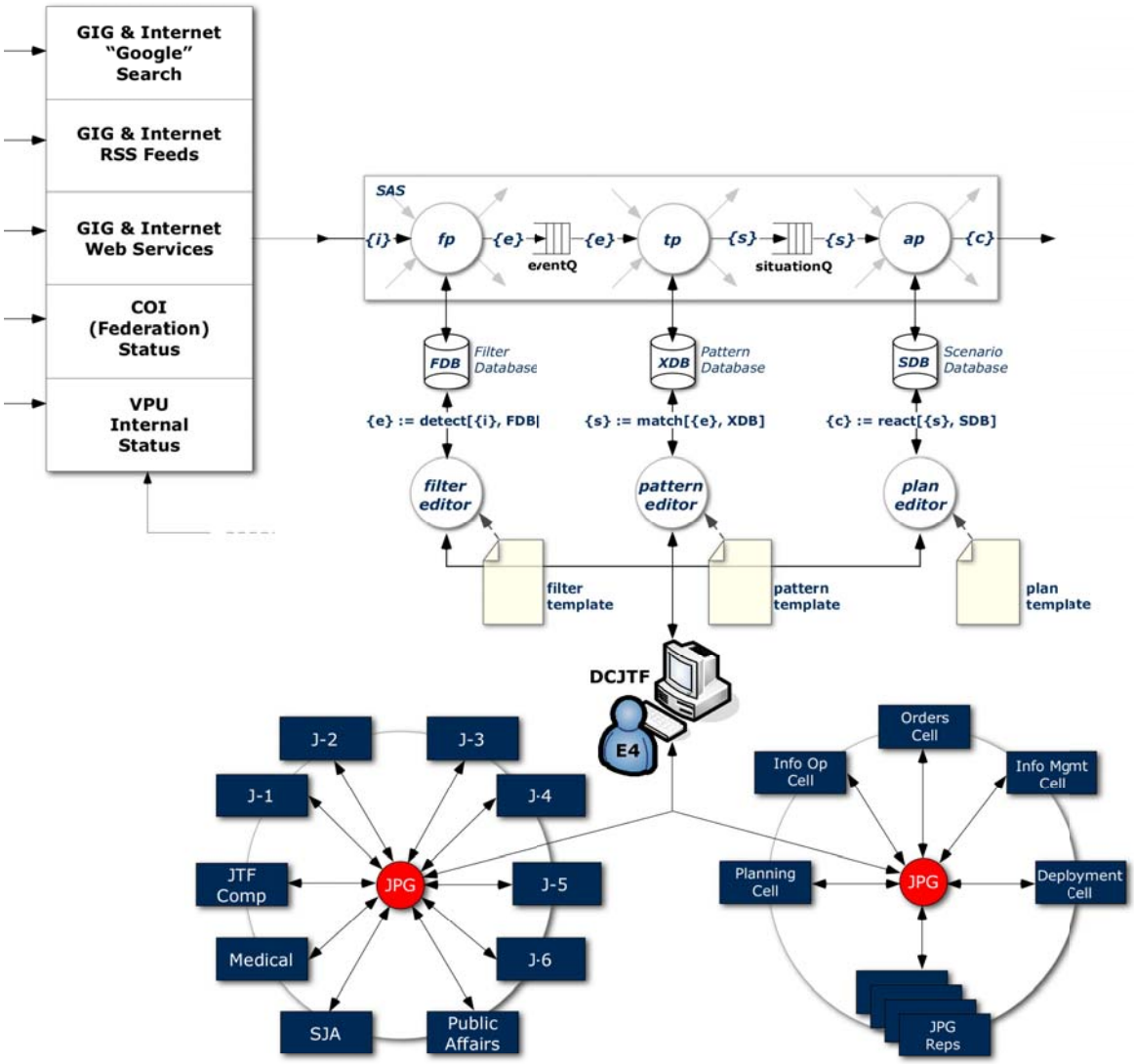

Figure 14 – JTF Situation Assessment Services

4.3.1.2 Listening

Given a number of active subscriptions (channels, feeds), an enterprise listens according to various topic-specific rules. Listening may be *directed* to identification of specific content in a specific stream. It may also be *undirected*, operating as a tape recorder for later processing or for generating an historical record. It may focus on the occurrence of periodically recurring strings, or conversely, be focused on the one-time occurrence of a specific (unique) string. Listening may seek to identify a pattern of randomly occurring strings within a single stream, or a pattern of strings occurring within some period in multiple streams.

There are two basic types of information input streams, *{i} – differentiated* (distinguished) and *undifferentiated*. Differentiated information is accompanied by *meta-data*, semantic information about the data in the stream. Interpretation of undifferentiated information is subject to greater ambiguity, making filtering and event detection significantly more challenging.

© 2006 All Rights Reserved

Effective listening may also require the ability to decipher encrypted streams. This is a critical requirement at the core of *information assurance* (IA) policies. While technically challenging and critical to the success of EC2 systems in the DOD, security is also an important matter in other local, state and federal agencies, in commercial-industrial settings and in patient privacy (aka, HIPAA) requirements in healthcare. Its importance is noted, but not discussed further in this paper.

4.3.1.3 Event Detection

Detection of patterns in streams is the function of *filters*, algorithms SAS invokes on the channels to which it holds active subscriptions. Detection of a pattern or sequence of patterns constitutes an *event {e}*. Events of high priority, or that declare a significant change in the state of some observed process, are considered *alarms*. Detection of alarms and events is a principal function of the SAS front-end service. Supporting event detection requires interactive services for defining, testing, deploying and maintaining libraries of filters, some that are subscription channel-specific, and some generic. The filter database (FDB) provides this repository.

With reference to Figure 14, the transformation for the SAS event-detection stage *{fp}* is:

$$\{e\} = detect[\{i\}, FDB]$$

{e} is a recognized event and FDB is the filter (event signature) database.

4.3.1.4 Event Correlation

Event detection in one or several active channels produces one or more event streams that are each information-rich and context sensitive. Patterns of events may also define important [meta-] events of interest. Such correlated (second-order) events may trigger alarms and events of their own, or may trigger the occurrence of a new *situation, {s},* or a significant change in an existing situation. As with filtering, SAS must provide interactive tools for the definition, testing, deployment and maintenance of pattern *correlators*, second-order composite filters. The pattern database (XDB) in Figure 14 serves this function.

4.3.1.5 Situation Detection

Correlators provide a means for detecting compound events of sufficient semantic richness to declare the occurrence of a *situation {s}* (i.e., an *incident*) or state change in a situation. Situation detection is the primary function of SAS. Incidents are the *meta*-events requiring a formal response (reaction) from the enterprise. SAS concludes by using situations (incidents) as indexes, or associative addresses, into a *situation database* (SDB). The SDB contains zero or more pre-existing plans (recipes) describing potential courses of action (COA) an enterprise may use to respond. If no prescribed response is present, either the event is ignored or a plan is created "on the fly" by the plan management ("PlanMan") application. If one or more plans are available, they are prioritized and sent to the PGS stage.

As implied by Figure 14, SAS must provide interactive tools for creating, testing, deploying and maintaining the filter, pattern and scenario database contents. The figures shows these interactive SAS stage editors for managing 1) network-hosted subscription filters, 2) pattern correlators and 3) plans (scenarios).

The transformation for the SAS situation-detection stage *(tp)* is:

© 2006 All Rights Reserved

$$\{s\} = assess[\{e\}, XDB]$$

{s} is a situation list, *{e}* is an event list and XDB is the pattern (situation signature) database.

4.3.1.6 Courses of Action

The first goal of SAS is to recognize significant events *{e}* and the situations *{s}* to which they apply. Its second goal is to suggest appropriate responses *{c}*, assuming the enterprise command authority had the presence of mind to anticipate the situations and to plan accordingly. Consequently, SAS is a function (service) whose inputs are information streams *{i}* and whose output is one or more planned courses of action *{c}*.

```
COA {
    plan_id;                    /* plan identifier */
    plan_revision;              /* plan revision level */
    plan_issue_date;            /* plan publication date */
    plan_mission;               /* plan objectives */
    plan_conops;                /* plan concept of operations */
    plan_tuf;                   /* plan TUF parameters */
    plan_assets {               /* plan contingencies */
        plan_personnel;         /*    human assets */
        plan_material;          /*    material */
    plan_precedents;            /* plan precursors */
    plan_metrics;               /* plan performance metrics */
    plan_policies;              /* plan constraints */
    plan_distribution;          /* plan distribution list */
    plan_task_list};            /* plan tasks */
```

Figure 15 - Generalized Courses of Action

Figure 15 is pseudo-code defining the general structure of a plan. In actual implementation, such COA might be defined as objects represented in a Scenario Database as XML Schema. COA are generalized response (reaction) plans to situations that have been seen in the past or are anticipated in the future. As noted in Figure 16, the SAS planning function is under the supervision of E4, the JTF DCJTF, and its management services are provided through the notional interactive JPG model management ("ModMan") and "PlanMan" applications.

4.3.2 JTF Plan Generation Services

With reference to Figure 16, the end-to-end transformation for the PGS stage is:

$$\{p\} = reserve[ADB[comply[\{c\}, PDB]]]$$

PGS is a two-stage process that takes as input COA *{c}* produced in SAS and produces as output plans of record *{p}*. POR are COA that have been validated as to policy compliance and have had their required resources assigned (reserved). A plan can fail at either of these two stages. We begin with compliance.

© 2006 All Rights Reserved

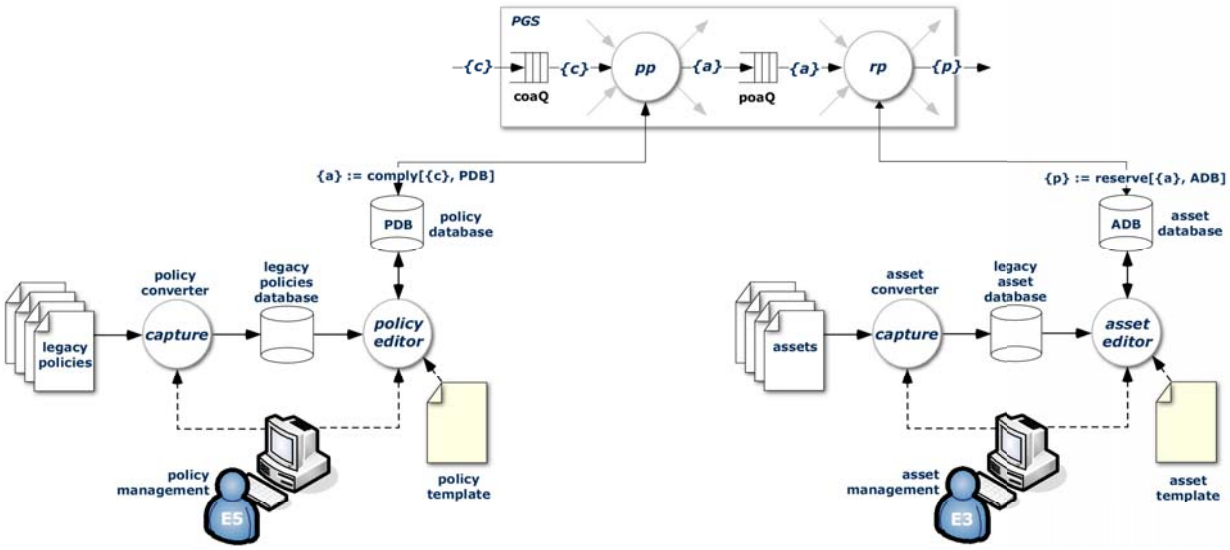

Figure 16 - JTF Plan Generation Services

4.3.2.1 Policy Management (Rules of Engagement)

With reference to Figure 16, the transformation for the PGS policy-compliance stage *{pp}* is:

$$\{a\} = comply[\{c\}, PDB]$$

{a} is a compliant plan of action, *{c}* is a proposed course of action and PDB is the policy database.

A policy is a formal guidance influencing the behavior of a managed resource or course of action. As a tangible object, a policy is text, written in a natural (e.g., English) or formal (e.g., XML encoded English) language. Figure 17 is an example of a generalized policy object, written in a semi-formal style. Again, the format shown is illustrative, not prescriptive and the specification is notional.

```
policy {
        policy_id;                      /* policy identifier */
        policy_revision;                /* policy revision level */
        policy_issue_date;              /* policy publication date */
        policy_end_date;                /* policy duration */
        policy_domain;                  /* policy application domain */
        policy_scope;                   /* policy applicability */
        policy_exceptions {             /* policy contingencies */
                policy_time;            /* vise time */
                policy_resources};      /* vise material */
        policiy_antecedents;            /* policy precedents */
        policy_authority;               /* policy owner */
        policy_clauses;                 /* policy if…then…else expressions */
        policy_distribution};           /* policy distribution list */
```

Figure 17 - Notional Policy Structure

If the policy service determines the COA to be compliant, it is forwarded to the resource process (service), *rp*. If it fails the policy compliance check, an exception is

© 2006 All Rights Reserved

raised and the commander is required to 1) address the policy deficiencies of the COA, 2) ignore the exception and assume responsibility for the risk(s) incurred, or 3) abort the plan altogether and either develop a new compliant plan or ignore the situation altogether.

COA arrive at the left *{c}* and enter a [priority] queue (coaQ) to await a policy compliance check. The check is automatic, calling upon a compliance service (comply[]). At the bottom of the figure a policy manager, in this case our illustrious E5 commander is responsible for developing and maintaining enterprise polices, by editing new and converting legacy policies.

The policy editor, prior to depositing a new or edited policy into the policy database (PDB), makes sure it is consistent with other policies and properly version controlled. The editor is also responsible for distributing new policies to addressees identified in their respective distribution fields.

The PGS policy process is a function (service) whose inputs are "raw" courses of action (COA) and, given policy compliance is met, whose output is a plan of action (POA) that is ready for asset allocation. The policy management function is identified as the policy management ("RuleMan") application.

4.3.2.2 Asset Management

With reference to Figure 16, the transformation for the PGS asset-allocation stage *{rp}* is:

$$\{p\} = reserve[\{a\}, ADB]$$

{p} is a resourced plan of record, *{a}* is a policy-compliant plan of action and ADB is the asset database.

An asset (ref. Figure 18) is any tangible resource required to carry out a plan. Assets (objects) may be people, material or capital. Assets may be consumable (e.g., money) or fixed and serially reusable (e.g., a warship). Material assets come from suppliers. Financial and labor (e.g., warfighter) assets are typically acquired from superiors.

```
asset {
    asset_id;                   /* asset identifier */
    asset_type;                 /* asset type */
    asset_deployment_date;      /* asset in_service date */
    asset_life;                 /* asset duration (exp) */
    asset_domain;               /* asset usage domain */
    asset_quantity;             /* asset quantity */
    asset_count;                /* assets in use */
    asset_qualifiers {          /* asset restrictions */
        asset_time;             /* asset use time restrictions */
        asset_capability;}      /* asset functionality */
    asset_prerequisites;        /* asset precedents */
    asset_authority;            /* asset owner */
    asset_usage;                /* asset SLA */
    asset_reservations;         /* asset reservations */
    asset_cost};                /* asset deployment & use costs */
```

Figure 18 - Notional Asset Structure

© 2006 All Rights Reserved

An enterprise may share its assets with members of one or communities of interest. Resource sharing is defined in mutual aid agreements (MAA), memoranda of understanding (MOU), service level agreements (SLA) or other such contracts. Resources not shared externally may be shared among the JTF's internal production units (P_n in Figure 6).

When a resource is shared, it is assumed serially reusable, utilized by a single JTF during any given period. Sharable resources require reservations. The resource process is the arbiter of resource requests for plans flowing through an enterprise and request by allied enterprises. Consequently, it must support a formal resource reservation protocol. Some resources, to be effective, require the presence of other resources (prerequisites). Resources may have restrictions by policy or by domain. In addition, resources have deployment and usage costs, generally denominated on dollars.

Figure 16 diagrams the PGS stage asset (resource) management service, *rp*. The rp service takes as input policy-compliant *plans of action {a}* and, to the extent resources are available, produces executable *plans of record* (POR), *{p}*. Resourcing a given plan is perhaps the most difficult yet critical service within EC2.

Assuming a JTF contains all necessary resources for all plans and all situations, the problem is easy. However, it is likely the heavy costs incurred for such an extreme resource level will prevent the JTF from being efficient or cost effective. Resource utilization will be low, capital costs will be high and value propositions will likely fail. For each JTF some "optimal" resource level will allow it to meet most of its objectives most of the time. In cases where a JTF operates at or near overload conditions, its resource management services are most critical in meeting operating commitments. This is precisely the situation our EC2 theory and its Utility Accrual Scheduling is focused.

Notice in Figure 16 the role of the asset manager, the JTF's DCJTF, the operations exec E3. The asset management service is identified as "AssetMan." E3 converts newly acquired and legacy assets into net-accessible resources. The editing process also guarantees that these resources, if sharable, are also schedulable. This implies that the *rp* service publishes the status of its asset inventory to a web service for use by allied enterprises who may wish to "lease" one or more of its assets.

We will discuss the critically important function of resource management in the following chapter – specifically, dynamic scheduling of resources to meet completion time requirements. This aspect of the EC2 theory is critical to achieving real-time or near real-time performance in enterprise systems. For the present, the asset management model in Figure 18 represents the static allocation service. Dynamic management of resources is another subject altogether.

In summary, the *(rp)* service is a function that maps policy compliant plans of action (POA) to fully resourced and ready-to-run plans of record (POR). What remains is for the commander (E5, CJTF) to authorize the plan. That requires completion of a concept of operations (CONOPS) to plan express "commander's intent," and to "fit" (i.e., schedule) the authorized plan into the running enterprise system.

At any given time, an enterprise is busy handling situations within the scope of its mission and area of operation. Accepting a new to will likely require it to reshuffle priorities and resources and in doing so to make adjustments and compromises in

© 2006 All Rights Reserved

activated plans. This is the nature of agile management. Moreover, the more complex an enterprise, the more difficult these adjustments are likely to be. This is the function assigned to the Plan Execution Stage (PES).

4.3.3 JTF Plan Execution Services

With reference to Figure 19, the end-to-end transformation for the PES stage is:

$$\{i\} = run[launch[authorize[\{p\}, CDB], TDB], \tau DB]$$

{i} is the output information produced from running the authorized plan of record *{p}*, CDB is the command database, TDB is the task execution performance database and τDB is the thread execution performance database.

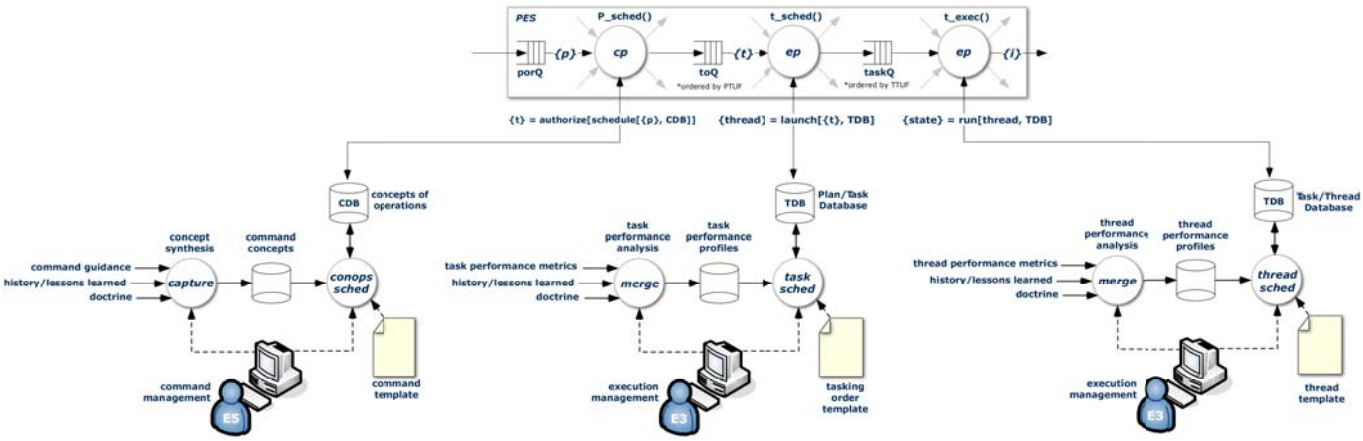

Figure 19 - JTF Plan Execution Services

EC2 Plan Execution Services (PES) provide for the scheduling, authorization and execution of plans of record (POR). The stage involves command processing *(cp)* and two stages of execution processing *(ep)*, one for medium grain (task-level) management and one for fine grain (thread-level[11]) management.

4.3.3.1 Scheduling

With reference to Figure 19, the transformation for the PES authorization stage *(cp)* is:

$$\{t\} = authorize[\{p\}, CDB]$$

{t} is an authorized tasking order, *{p}* is a resourced and compliant plan or record and CDB is the command database.

Successful execution of plans is highly dependent on correct sequencing of the time-dependent tasks they perform and on the availability of resources that they require. Effective scheduling of tasks and resources is therefore crucial to effective management. In this sense, scheduling is a complex subject, beyond the scope of this document. However, several key concepts are important to this presentation.

Scheduling, involving timing and resource requirements, occurs at several successive stages along the CPS chain. Initial (default) schedules and resource requirements

[11] A *thread* is a manageable (schedulable) task step or sequence of task steps within a plan.

© 2006 All Rights Reserved

appear in COA specifications contained in the scenario database (SDB). Plan timing requirements derive from initial plan and task level completion time requirements, expressed as *time-utility functions* (TUF) [53-56]. Plan resource requirements derive from initial plan and task level resource specifications.

Schedules are adjusted during *(rp)* stage resourcing activities. Here, resource requirements are identified and a reservation for each required resource is secured according to the TUF specifications. Some reservations may (typically will) require adjustments to the task TUFs to account for resource availability, acquisition and release times.

The resulting schedule, as defined in the POR produced by *(rp)*, defines a "feasible plan," but one that may not be "optimal" in terms of resource utilization, enterprise accrued utility, or side-effects of plan execution (so-called "execution byproducts"). As such, final judgment of the effectiveness of the scheduling activity rests with the JTF commander (E5). This judgment is one of the key concerns of the PES stage, and a determination that naturally must precede plan authorization and execution. However, before this determination is made, the commander must have a clear and concise notion of the JTF's mission and the particular role the plan has in its accomplishment. That, in turn, requires clear and concise concepts of operations (CONOPS).

4.3.3.2 Command Structure

Figure 20 is a notional structure describing the core elements of the command context.

```
command {
    command_id;                  /* command [URI] identifier */
    command_authority;           /* command authority level */
    command_name;                /* command [string] name */
    command_size;                /* N, number of domains */
    command_aor[N];              /* per-domain areas of responsibility */
    command_start;               /* per-domain commission date */
    command_end;                 /* per-domain decommission date */
    command_hq[N];               /* per-domain HQ location */
    command_exceptions {         /* per-domain contingencies */
        command_time;            /* vise time */
        command_resources};      /* vise material */
    command_superiors[N];        /* per-domain superiors */
    command_subordinates[N];     /* per-domain subordinates */
    command_suppliers[N];        /* per-domain suppliers */
    command_clients[N];          /* per-domain clients */
    command_por[N,M];            /* per-domain POR */
    command_conops[N,M]};        /* per-plan CONOPS */
```

Figure 20 – Notional Command Structure

4.3.3.3 Concept of Operations

Authorization involves two primary steps: 1) developing a concept of operations (CONOPS) for the plan and 2) developing a feasible schedule for insertion of the plan into the JTF's mix of currently executing plans.

© 2006 All Rights Reserved

A concept of operations (CONOPS) specification is required for each plan of record presented to the commander for authorization. The CONOPS provides a clear and concise expression of the commander's intent related to the more detailed plan, and includes a summary of objectives, timing, resources, positions, maneuver, logistics and contingencies of the POR (e.g., battle plan). Production of the requisite CONOPS documentation is performed using a standardized template, as indicated in the figure.

4.3.3.4 Plan Scheduling

Following development of an appropriate CONOPS for the pending POR and incorporation of any potential adjustments to timing or resources, the commander is free to "test" the viability of plans against JTF capability. This test results, in general, in one of three outcomes.

1. The plan is executable as described, with an expected utility[12] U1
2. The plan is executable as described, but given resource levels, it is expected to complete late with utility U2<U1
3. The plan is not executable as described, but with the following recommended adjustments in time (TUF parameters) can be made executable with utility U3; time adjustments will affect the following in process plans *{…}*
4. The plan is not executable as described, but with the following adjustments in resources (reservations) can be made executable with utility U4; resource adjustments will affect the following in-process plans *{…}*

Ideally, *(cp)* services, with aid of the command management ("SuperMan") application, are able to answer the following query:

> *"What optimal schedule(s) result from adding this new plan (POR) into the mix of in-process plans and that produce the maximum utility for my JTF?"*

If such a query produces one (or more) optimal schedules, the required subset of the current mix of in-process plans are aborted and rescheduled, including the pending one, adjusting them to the optimal schedule; they are then re-launched.

A useful variation on the query might be:

> *"What optimal schedule(s) result from adding this new plan (POR) into the mix of in-process plans and result in an overall utility between U_{lower} and U_{upper} bounds?"*

4.3.3.5 Plan Authorization

Given that *(cp)* is successful in finding a feasible coarse-grain schedule meeting utility objectives, the next formal act of a command is to "authorize" the pending POR. This is, in effect, affixing an electronic "signature" to the POR and issuing it as a tasking order to the JTF's *execution (ep)* stage, presumably with "copies" to the JTF's superior and all affected federation peers.

---

[12] The term *utility*, as used here, is a performance measure based on *time-utility functions* (TUF) and the result of a computation referred to as the *utility accrual* (UA) algorithm. Discussion of the concept of utility and the UA algorithm may be found at http://www.real-time.org. Also see [20] at http://www.echelon4.com/references.htm.

© 2006 All Rights Reserved

Prior to signing a POR the command [staff] may wish to "validate" that the plan is still "executable." Such scenario testing has been the subject of considerable research, with several tools available for restricted use.

4.3.3.6 Plan Execution

Again, with reference to Figure 16, the transformation for the PES task execution stage *(ep)* is:

$$\{\tau\} = launch[\{t\}, TDB]$$

{ τ } are the task threads resulting from launching and executing the tasking order *{t}* and TDB is the task performance database.

Upon receipt of a new tasking order, the E3 actor has the job of overseeing execution of the plan and its elemental tasks according to specifications contained in the order and in compliance with the JTF commander's (E5) intent, as expressed in the accompanying CONOPS. The received tasking order has the notional form described in Figure 21, below, a structure that mimics the plan structure previously defined in Figure 20.

```
tasking_order {
      to_id;                    /* to identifier */
      to_revision;              /* to revision level */
      to_issue_date;            /* to publication date */
      to_mission;               /* to objectives */
      to_conops;                /* to concept of operations */
      to_tuf;                   /* to TUF parameters */
      to_assets {               /* to contingencies */
          to_personnel;         /*    human assets */
          to_material;          /*    material assets */
      to_precedents;            /* to precursors */
      to_metrics;               /* to performance metrics */
      to_policies;              /* to constraints */
      to_distribution;          /* to distribution list */
      to_task_list};            /* tasks */
```

Figure 21 - Notional Tasking Order (TO) Structure

5 JTF Performance Considerations

To assess the effectiveness of a JTF or a federation of collaborating enterprises, the JTF ECS model includes a performance measurement framework (PMF) that includes two classes of metrics (ref. Figure 21, "to-metrics"). The first class includes measures that define a JTF's performance in terms of its capability and actual performance. The second class of metrics defines a JTF's *utility*, its ability to meet plan completion-time requirements. Performance defines throughput (task completion rate); utility defines the value of the task to the overall mission. Both classes are required to scale vertically along the command axis and horizontally along the production axis.

5.1 JTF Performance Metrics

Figure 24 defines the JTF basic performance indices. Table 5 summarizes the primary performance indices.

© 2006 All Rights Reserved

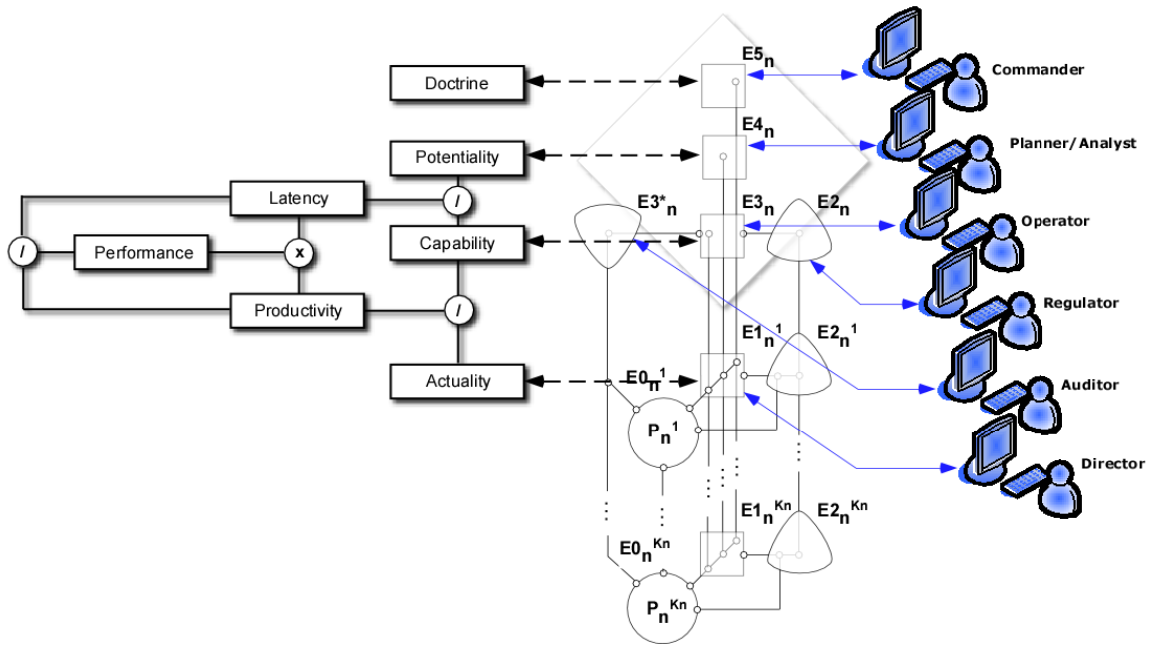

Figure 22 - JTF Performance Indices

Metric	Function
Potential	Design (architectural) limits
Capability	Deployed (e.g., funded) level of capacity
Actuality	Actual (instantaneous) performance
Latency[13]	Latent potential (unused, unfunded) design potential
Productivity	Utility of deployed (used, funded) capability
Performance	Absolute performance of the enterprise as designed

Table 5 - JTF Performance Indices

5.2 JTF Utility Measures

Utility, while related to the performance indices introduced above, is an important but distinct metric. Utility is a measure of the value of a task as a function of its completion time [53, 55, 56]. Utility, of a task, a plan, or a set of plans depend on the value of completion by a deadline, early or late. Figure 23 diagrams several such value propositions. Here, the utility of a task or plan is plotted versus its completion time.

For a JTF command responsible for the overlapping execution of several tasks (plans), overall utility is a form of rolling sum (e.g., convolution integral) of the utilities of all active tasks (plans) [64]. The algorithm that computes this integral is based upon the utility accrual (UA) model. Together, these two metrics define parameters for guiding a commander in simultaneously achieving "commander's intent" while maximizing performance and accrued utility. Figure 24 diagrams the concept.

[13] As used here, the term *latency* refers to the capability latent or unused in a system, not the usual communications-oriented notion of temporal delay, lateness or tardiness of an event.

© 2006 All Rights Reserved

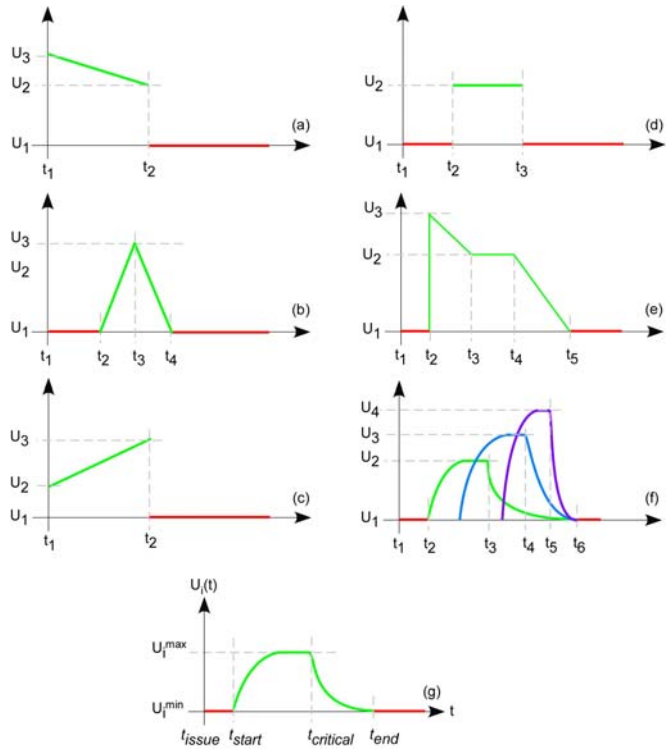

Figure 23 - Example Time-Utility Functions

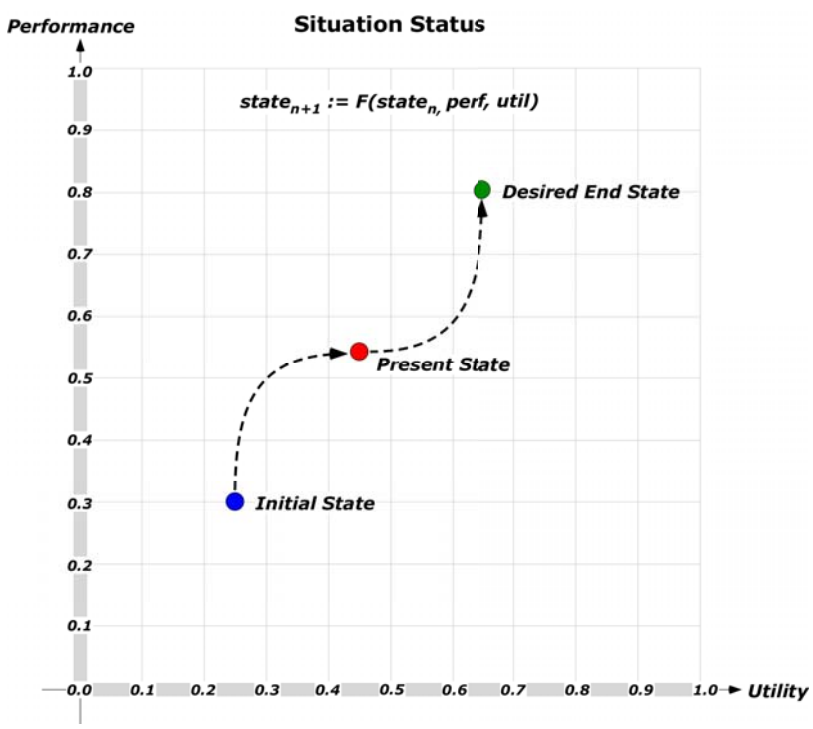

Figure 24 - JTF Performance

© 2006 All Rights Reserved

6 Conclusions

The JTF EC2 model provides improvements in the conceptualization, structure, function and scalability of JTF C2 concepts as defined in Joint Publication 5-00.2. Furthermore, it extends the JTF concept by recasting it in a services-oriented and network-centric framework, with the goal of defining a service-oriented software *framework* applicable to next generation (i.e., post GCCS/JOPES) JTF implementations.

Multi-agency (multiple JTF) communities of interest (tactical federations) are of particular interest, motivated by requirements for coordinated governmental and non-governmental responses to terrorist and natural disaster detection, response and recovery. Collaborative federations of JTF enterprises demand C2 systems and services that are competent to handle internal and inter-agency situation assessment, plan generation and plan execution – in short, establishment of a "C2 commons."

Finally, we have endeavored to extend the JTF concept by explicitly introducing command (vertical) and logistics (production) axes on which it necessarily must operate and redefining the control processing services (CPS) required to manage the information flows that enter and exit along these axes. Control processing defines the core network-centric services to be shared among collaborating command enclaves and their staffs.

© 2006 All Rights Reserved