SSC99-VI-4

Discrete Event Command and Control for Formation Flying of Distributed Small Spacecraft Systems

P. A. Stadter, Ph.D.

The Johns Hopkins University
Applied Physics Laboratory
Laurel, MD 20723-6099
patrick.stadter@jhuapl.edu

Abstract. A distributed, multi-vehicle concept for future space missions has been conceived as a solution to the problem of advancing space-based operations within budgetary constraints. Broadly named formation flying, this approach to designing distributed systems across multiple, spatially disbursed platforms is enabled by collectively coordinating a fleet of autonomous spacecraft to function as a unified system. Formation flying offers potential advantages of improved robustness, capability, and cost relative to complicated, single platform systems by using multiple, often small, spacecraft to perform complex multi-sensor tasks. A necessary element in the realization of formation flying systems is the development of methods and technologies that facilitate the transition from treating a distributed spacecraft system as individual elements, to viewing a formation as a coordinated system unified by common objectives.

This paper describes the results of research performed to identify fundamental issues that affect the development of command and control (C^2) methods applicable to coordinating distributed small spacecraft systems. A discrete event method of distributed command and control is described that is particularly well suited to small spacecraft formation flying. Utilized in many complex terrestrial systems, discrete event systems (DES) concepts facilitate coordination of distributed systems at multiple levels of resolution in an efficient manner. DES also provide a means to integrate intelligent planning and processing operations while interfacing with more traditional subsystem controllers. The basic principals and applicability of DES are described within the context of formation flying and example distributed spacecraft C^2 operations are defined.

Introduction

Formation flying captures the concept of implementing space campaigns using multiple, distributed spacecraft controlled as a coordinated, semi-autonomous system.[26,4,28,29] This approach to space missions is based upon the distribution of capability among multiple, individually limited spacecraft whose collective faculties result in distributed systems applicable to co-observation, multi-point observation, and other complex, multi-sensor tasks.[18] The basic goal of formation flying is to achieve a transition from controlling individual spacecraft to interacting with distributed autonomous spacecraft as a coordinated system unified by common objectives. Methods and technologies that facilitate this transition provide potential advantages in distributed spacecraft system design and operation in the form of increased capability, improved robustness, and reduced complexity and cost relative to nominally equivalent single-spacecraft systems.

Given these potential advantages, there is a tight coupling in terms of desired system characteristics between formation flying and many small satellite programs. This coupling is evident in the fact that many small satellites are designed to achieve mission objectives with limited resources and within tight budgetary constraints. Indeed, due to the fact that distributed spacecraft systems rely upon the collective capability of multiple assets, cost constraints often necessitate that small, economical spacecraft represent those assets. Within the context of small satellite programs, formation flying can contribute a formal methodology for harnessing the capabilities of individual spacecraft while supporting system robustness through gradual degradation of capability in the presence of system faults or asset loss.

A fundamental driver for the development of distributed spacecraft systems and associated formation flying technology are the aggressive scientific and technological goals of many space campaigns that are envisioned for the future. These missions typically rely on the use of multiple sensors separated beyond the range of a single spacecraft to perform system operations such as multi-point observation, distributed aperture reconnaissance, and space-based interferometry. Specific applications include NASA's Geospace Multiprobes,[10,18] which will use multiple highly coordinated satellites to explore Earth's upper atmosphere and space environment and the Air Force's TechSat 21 program,[8] which will perform surveillance using distributed radar sources. Proposed near term missions focus upon formation flying technology demonstration, coupled with modest scientific goals. Two such technology demonstration programs with a small satellite focus are the NASA Orion mission[11] and the NASA/DoD University Nano-Satellite Program. These missions focus upon the demonstration and refinement of fundamental formation flying concepts such as distributed control and interspacecraft communications.

The technological advances in formation flying necessary to support these applications require addressing issues inherent in the development and operation of distributed, autonomous systems. This includes scalability to support the dynamic augmentation of assets, design flexibility, operational dynamism to provide real time adaptivity, and efficiency in the sense of restrained computational, physical, and operational costs. These issues act as constraints during system design, and because many proposed systems envision the use of individually limited, small spacecraft, efficiency is notably important.

The research described herein presents an approach to coordinating the assets of multiple spacecraft systems that is particularly well suited to small spacecraft formation flying. Based upon discrete event systems (DES) concepts, this approach to distributed command and control (C^2) facilitates needed efficiency while providing the scalability, flexibility, and adpativity required for system implementation. Used in a variety of complex, terrestrial applications, DES control approaches support coordination at multiple levels of resolution in an efficient manner and provide a means to integrate intelligent planning and processing operations while interfacing with traditional subsystem controllers.[7,20,21,12] In addition to a discussion of the basic aspects of DES as they relate to formation flying, consideration is given to fundamental architectural issues that affect implementation. This includes the functional relationship among distributed system assets and the interspacecraft communications infrastructure that supports C^2 operations.

Discrete Event Command and Control

Control of distributed spacecraft systems in formation flying applications requires novel coordination techniques that support the highly complex, autonomous nature of spatially disbursed system components (e.g., spacecraft) while operating within challenging, unpredictable environments. Traditional control techniques that seek to physically model a plant to be controlled can represent an infeasible solution when applied to distributed, autonomous systems that can be characterized as nonlinear, time varying, and stochastic.[13] In particular the behavior of individual autonomous components and the emergent interactions that occur among them generally defy modeling through standard differential or difference equation techniques. This complexity is not restricted to formation flying applications, and indeed can be seen in numerous terrestrial systems such as manufacturing processes, shipping networks,[7] combat dynamics;[19,24] networks of autonomous vehicles,[21,15,14] and automated intelligent highways.[12]

While conventional control techniques focus on meeting requirements in terms of typical parameters, such as response time, stability, and robustness, the increasing complexity of systems to be controlled has necessitated the development strategies that address augmented requirement sets. These augmented requirements include dynamic system behavior adaptation, coordination of autonomous system assets, goal planning and refinement, fault resolution, and learning.[15,9] This has led to the development of *intelligent control* techniques, which seek to control the behavior of complex systems through empirical observation and by interfacing with conventional controllers rather than by evolving a detailed model of the plant. Numerous techniques have been applied to the implementation of intelligent control systems, including neural networks, fuzzy logic, discrete event systems, ontologies, and expert systems.[3] While each of these has inherent advantages in terms of complex system control, discrete event systems concepts provide broad capabilities and leverage particularly powerful formalisms for the control of multiple, autonomous entities, such as satellites in formation flying.

DES are dynamic, discrete time systems that can be represented by a discrete state space and a state transition structure. The state space consists of all possible configurations that the state of the system can assume, and the state transition structure defines the mechanisms by which a system evolves within its state space. For DES the state transition structure consists of a set of events, which captures physical occurrences that drive state transitions; these discrete events are typically assumed to be instantaneous, asynchronous, and nondeterministic.

The inherent characteristics of DES concepts provide a powerful array of tools for the modeling and control of complex, autonomous systems. These characteristics include:

- The ability to model systems or system components at multiple levels of abstraction; the granularity of the state space and the event set can be designed to be sufficient to meet modeling requirements.
- The capability of defining and implementing control methods for complex systems that resist characterization by conventional control methods.
- The facilitation of hierarchically distributed control structures that allow modular, replicable designs, lead to natural task decomposition, provide methods to address complexity and scalability issues, and support information aggregation.
- The capability of interfacing with conventional control techniques as well as inferencing technologies (e.g., neural networks, fuzzy logic) to track the state of the system, identify event occurrences, and affect control actions.
- The natural coupling with well studied, formal modeling and representation techniques such as finite state automata, Petri nets, and formal languages.

Discrete Event System Fundamentals

Effective representation of DES has been achieved through the use of finite state automata (FSA) and associated formal language theory constructs.[25,33] While there is no theoretical limitation necessitating a finite state representation, most practical implementations do not require a state set of infinite cardinality. Using an FSA representation the DES is formally modeled as a 5-tuple:

$$R = (Q, \Sigma, d, q_0, Q_m) \text{ where} \tag{1}$$

Q is a set of states, possibly infinite,
Σ is a finite set of events,
called the alphabet,
d is a partial mapping from $Q \times \Sigma \rightarrow Q$,
denoting state transitions,
q_0 is the starting state, and
$Q_m \subseteq Q$ is a set of marked states.

Denote by Σ^* the set of all finite strings of elements from Σ, including the empty sting *e*. Using the notation $d(q,s)!$ to indicate that $d(q,s)$ is defined, the transition function is inductively extended to strings as $d(e,q)$ and $d(s\,s\,,q) = d(s\,,d(s,q))$, whenever $q' \overset{\Delta}{=} d(s,q)!$ and $d(s\,,q')!$. Any $K \subseteq \Sigma^*$ is a *language*, and

$\overline{K} = \{s \in \Sigma^* | \exists q \in \Sigma^* \ni sq \in K\}$ is the *prefix closure* of *K*. Therefore the language associated with *R* is

$$L(R) = \{w | w \in \Sigma^* \text{ and } d(q_0, w)!\}, \qquad (2)$$

And the language marked by *R* is

$$L_m(R) = \{w | w \in L(R) \text{ and } d(q_0, w) \in Q_m\}. \qquad (3)$$

This formal model of a DES operates by starting in the initial state and executing state transitions as defined by d and given a finite sequence of events from the alphabet Σ. If the DES is in a marked state after executing a string of events, the DES *recognizes* the string; $L_m(R)$ therefore represents the set of all finite length strings that are recognized by *R*. Qualitatively, marked states distinguish those strings that have significance to the operation of the DES, such as the completion of a task or the attainment of a goal.[33,19]

Example 1. This example is based on the mutual exclusion/fair usage development in Ramadge.[25] Consider three spacecraft flying in formation and responsible for reconnaissance operations. An important aspect of such a mission is verifying the health of the spacecraft assets to predict and monitor potential system liabilities that may impact operations supported by the distributed spacecraft system.

Assume that one spacecraft (S/C0) will make contact with a ground station to download observations and formation health. It is necessary for the two remaining spacecraft (S/C1 and S/C2) to communicate their current system status to S/C0, requiring crosslink data exchange. This scenario can be modeled as a mutual exclusion/fair usage problem under the following assumptions: only one spacecraft can exchange data on the crosslink at a time, and crosslink access is granted to requesting spacecraft on a first-come, first-served basis.

A DES model for this scenario is identical for S/C1 and S/C2, and it is given by $R_{S/Ci}$, $i = 1,2$:

$$R_{S/Ci} = (Q_i, \Sigma_i, d_i, q_{0i}, Q_{mi}) \text{ where}$$

$$Q_i = \begin{Bmatrix} \text{quiet crosslink, request crosslink,} \\ \text{exchange S/C health} \end{Bmatrix},$$

$$\Sigma_i = \begin{Bmatrix} a_i : \text{request crosslink exchange} \\ b_i : \text{crosslink exchange granted} \\ c_i : \text{crosslink terminated} \end{Bmatrix},$$

$$q_{0i} = \{\text{quiet crosslink}\},$$

$$Q_{mi} = \{\text{quiet crosslink}\}, \text{ and}$$

δ_i, specifying state transitions, is defined in Figure 1.

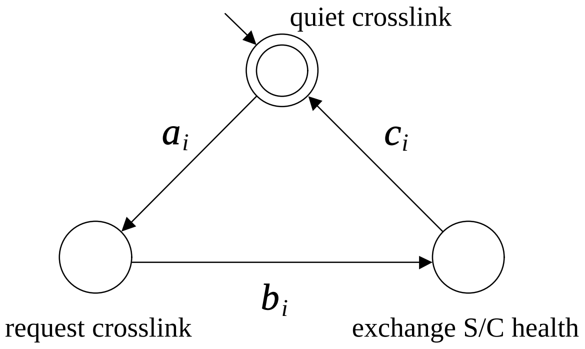


Figure 1. State Transition Diagram for Crosslink Exchange of Spacecraft Health and Status.

Each spacecraft begins a health and status update cycle in the {quiet crosslink} state, as indicated by (→) in Figure 1. Note that this is also a marked state, which is denoted by concentric circles in the state diagram. When the spacecraft requests a crosslink exchange with S/C0, A transition to the {request crosslink} state results when S/C*i* requests a crosslink exchange with S/C0. When that request is granted, S/C*i* enters the {exchange S/C health} state, and returns to the {quiet crosslink} state when the exchange is terminated.

Control of a DES *R* can be realized by the development of a supervisor $T = (T_a, j)$ that consists of an FSA, T_a, augmented by an appropriate state feedback map j. This requires an alternative view of the DES *R* in Eq. (1). Specifically, in addition to acting as a recognizer of strings $s \in L_m(R)$, *R* can be viewed as a *generator*, which begins in the start state and nondeterministically selects an event that is generated as an output symbol as the DES transitions to the appropriate state based on the state transition function. The operation of *R* under the supervision of *T* is denoted by the DES *T/R* and is shown in Figure 2.

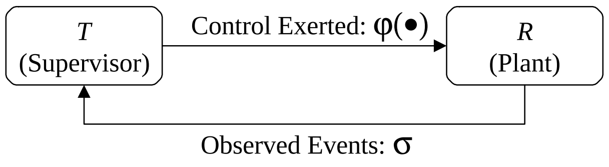


Figure 2. Closed Loop Control of a Discrete Event System.

Control is realized by modeling supervisor $T = (T_a, j)$ as a recognizer with state feedback map j, and DES *R* as a generator. Thus, referring to Eq. (1), $T_a = (X, \Sigma, x, x_0, X_m)$ and $j : X \rightarrow 2^{\Sigma}$. where 2^{Σ} represents the power set of the alphabet Σ. The set Σ can be partitioned into subsets of controllable events (Σ_c) and uncontrollable events (Σ_u) such that $\Sigma = \Sigma_c \cup \Sigma_u$. A controllable event is an action or occurrence that can be disabled, while an uncontrollable event cannot be affected by supervisory actions and is therefore considered to be enabled at all times. Conceptually, as *R* generates symbols in response to the occurrence of events, supervisor *T* performs state transitions, which cause the state feedback map j to produce a set of events that are enabled. The supervisor effectively enables and disables controllable events in the recognizer *R*, thereby restricting or allowing state transitions by the subordinate DES.

Example 2. Continuing the formation flying modeling of mutual exclusion/fair usage presented in Example 1, a DES model $T = (T_{S/C}, j)$ of a supervisor to coordinate the crosslink health and status data exchange among three spacecraft is developed. The supervisor includes an FSA $T_{S/C}$ developed by taking the shuffle [25] of the DES modeled in Example 1 by generators $R_{S/C1}$ and $R_{S/C2}$. The shuffle represents a process that models the concurrent actions of $R_{S/C1}$ and $R_{S/C2}$ under the assumption that those actions are asynchronous and independent.

The FSA $T_{S/C}$ is given by:

$$T_{S/C} = (X, \Sigma, x, x_0, X_m) \text{ where}$$
$$X = \{0,1,2,3,4,5\},$$
$$\Sigma = \Sigma_{S/C1} \cup \Sigma_{S/C2} = \{a_1, b_1, c_1, a_2, b_2, c_2\},$$
$$q_{0i} = \{0\},$$
$$Q_{mi} = \{0\}, \text{and}$$

x provides the state transitions as defined in Figure 3

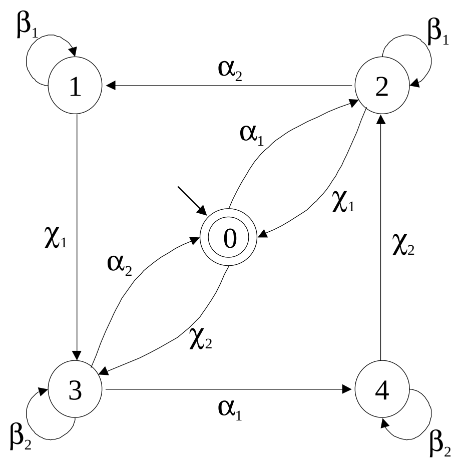


Figure 3. State Transitions for Supervisor *T*.

In conjunction with the FSA $T_{S/C}$, the feedback map φ must be specified. For this example,

$$j(x) = \begin{Bmatrix} b_i & \text{iff } b_i \text{ issues from } x \in X, i = 1,2 \\ \varnothing & \text{else} \end{Bmatrix}.$$

That is, if the supervisor *T* enters a state from which the event b_i can cause a transition, then the event b_i is enabled by φ. Note that the mutual exclusion assumption guarantees that only one b_i, *i=1,2*, emanates from any one state.

The DES control concepts described previously have been extended to incorporate multilayered, hierarchically modeled systems.[19,20,21] This provides a significant advantage in controlling highly complex systems because of the natural decomposition of the system structure. This can also mitigate scalability problems found in distributed, autonomous systems such as formation flying spacecraft. Specifically, hierarchical structures can be designed to support replication and reuse of control methods, particularly in systems that consist of multiple homogeneous assets. Another important advantage of a hierarchical DES control approach is that it directly supports the intelligent control concept of increasing intelligence at higher layers and increasing fidelity at lower layers. This is reflected in hierarchical DES control by information aggregation, in which the languages of higher level supervisors may be represented by multiple symbols from lower levels of the hierarchy. This is consistent with many realistic systems (e.g., military organizations) in which high levels are concerned with broad actions and planning, leaving the details of achieving those plans to subordinate layers.

Following the development in Peluso,[20] extension of the DES control model described previously requires the introduction of an augmented automaton that can act as both a recognizer and a generator. This augmentation is defined as a transducer $S = (S_a, j_S)$, which is modeled as a 6-tuple with the appropriate state feedback map for control:

$$S_a = (X, \Sigma, \Gamma, x, x_0, X_m) \text{ and } j_S : X \to 2^{\Sigma}, \qquad (4)$$

where X, x_0, and X_m are defined as previously specified, Γ is the alphabet of symbols generated as output, and $x : X \times \Sigma \to X \times (\Gamma \cup \{e\})$. A general representation of the information flow of a three-level DES control hierarchy appears in Figure 4. Each generator R_i is modeled as $R_i = (Q_i, \Sigma_i, d_i, q_{0i}, Q_{mi})$, each supervisor $S_i = (S_{ai}, j_{Si})$ is modeled as a transducer $S_{ai} = (X_i, \Sigma_i, \Gamma_i, x_i, x_{0i}, X_{mi})$ with state feedback map $j_{Si} : X_i \to 2^{\Sigma_i}$, and the top layer supervisor $T = (T_a, j_T)$ is modeled as a recognizer $T_a = (Z, \Gamma, t, z_0, Z_m)$ with state feedback map $j_T : Z \to 2^{\Gamma}$. The alphabets Σ_i and Γ_i are assumed to be disjoint, and $\Sigma = \Sigma_1 \cup \Sigma_2 \cup \cdots \cup \Sigma_n$ and $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \cdots \cup \Gamma_k$.

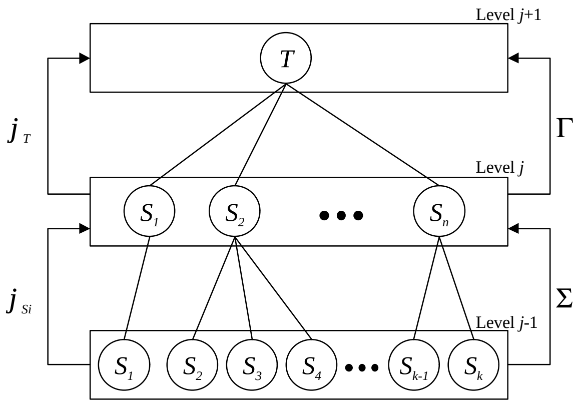


Figure 4. Information Flow in a Three Level Hierarchy.

In Figure 4 Σ is the set of symbols generated by R_i, $i = 1 \ldots n$, of level j-1 and recognized by transducers S_i, $i = 1 \ldots k$, of level j. Similarly, Γ is the set of symbols produced by the transducers of level j and recognized by T of level j+1. Control is exerted through the state feedback maps: j_T maps the states of T in level j+1 to subsets of the events recognized by S_i in level j, and j_{Si} maps the states of S_i in level j to subsets of the events recognized by R_i in level j-1. This hierarchical control process is achieved as follows:

- As a generator R_i in level j-1 outputs a symbol, its associated jth level supervisor S_i responds by making a state transition and exerting control via j_{Si}, enabling and disabling events in Σ.
- If the state transition of supervisor S_i generates an output symbol from Γ, the j+1 level supervisor T responds by making a state transition and enabling or disabling events in Γ via j_T.
- Generators are prohibited from generating symbols in Σ that are disabled by their associated supervisors S_i as well as any event in Σ that would cause a disabled symbol in Γ to be generated.

A fundamental advantage to modeling and controlling complex systems with DES techniques is the formal structure that exists. This has resulted in the development of DES control concepts analogous to results in conventional control theory. This includes formal definitions of stability, controllability, and observability as well as methods of supervisor synthesis.[20,15,16] Despite the ability to define DES models at multiple levels of abstraction, achieving fidelity sufficient to represent critical system operations and exercise intelligent control generally results in computationally intensive designs and implementations of DES control algorithms. Recent research has explored alternative DES control formalisms in an effort to reduce the computational complexity involved in controller design.[15] In addition alternative DES models have been developed in the context of hierarchical control, including learning stochastic automata,[17] Petri nets, and Markov chains.[7]

DES Architectural Considerations and Integration with Inferencing Technologies

Discrete event command and control of formation flying systems requires implementation that supports significant autonomy among spacecraft. The operational environment for formation flying generally precludes direct physical interaction with the systems and communications contacts are limited by system dynamics and communications constraints. For distributed small satellite programs in particular, resource limitations and the expense of ground support motivate the development of spacecraft autonomy.

Support for autonomous operations within distributed spacecraft systems is facilitated by recent efforts in the development of intelligent control architectures.[3] Generally focussed upon the control and coordination of highly autonomous, distributed assets, many intelligent control architectures are suitable for the integration of discrete event components to address nonlinear or ill-defined plants that resist modeling and control by conventional methods. One such architecture, developed by Saridis,[27] consists of a three level hierarchy: a top level knowledge-based organization level responsible for composing tasks from primitive operations, a coordination level responsible for dispatching tasks among system assets, and an execution level responsible for interfacing with the system environment through sensors and actuators. Another architecture proposed for complex systems control and applied to robotic manufacturing systems is the *reference model architecture*, developed by Albus.[1] Built upon the evolution of the Real-time Control System,[2] this approach to intelligent control consists of a hierarchically layered set of processing nodes that decompose tasks, generate plans, maintain world models, and process sensor feedback. As with most intelligent control methods, higher levels display a broader “world view” and function on different time scales than the more precise and limited operations of lower levels.

The *subsumption architecture*, developed by Brooks, is a behavior-based approach in which different agents compete to satisfy basic system needs.[6] Agent coordination is achieved through prioritization in which lower priority behaviors are subsumed by more important behaviors. Building specifically upon research in DES, Kumar and Stover recently presented the *behavior-based intelligent control* (BBIC) *architecture*.[13,14] This hierarchical control structure consists of explicit input and output interfaces between which function a perceptor and a response module. The perceptor extracts relevant information from sensor assets to construct an internal representation of the external environment, while the response controller determines control actions by evaluating goal achievement, constructing operational plans, and selecting system behaviors. Within this context behaviors represent independent operations required of the system to execute subtasks which comprise a basic mission task.[13]

The BBIC architecture is particularly attractive for the command and control of distributed spacecraft because of its heritage in modeling equivalently complex systems in the form of autonomous underwater vehicle networks.[22,23] In addition the BBIC architecture exhibits well-defined interfaces between the hybrid (continuous/discrete) control modules and the external system environment. This characteristic is of fundamental importance to intelligent control methods in general, because the control mechanism within each autonomous component of a distributed system must be capable of coordinating the actions of the individual component within the context of the overall system goals. The intelligent controller must also logically integrate inferencing capabilities to support decision-making. This integration often takes place at system interfaces, such as between component sensors and controller perception or world-view modules, or between intelligent controller decision functions and effector subsystems that implement those decisions. When incorporating discrete events systems into an intelligent controller, these interfaces may also represent the boundary between the continuous environment and the discrete control space. Techniques such as structured neuro-fuzzy networks and inferencing networks have proven to be effective solutions in these situations.[31,30,32]

The incorporation of inferencing techniques can be seen explicitly in Figure 5, which represents a high-level view of the BBIC architecture developed for the command and control of a network of distributed autonomous underwater vehicles. Figure 5 shows the intelligent control architecture for a single vehicle that represents an autonomous asset within a network of similar vehicles. Of particular note is the interface between the response controller and the effector subsystems that implement control actions. This supports the notion that the discrete intelligent controller decisions can act as set-points for conventional controllers.[13] Within a distributed spacecraft system this may be realized as a command, either generated autonomously within a spacecraft or commanded by a higher-level supervisory spacecraft, to adjust attitude in anticipation of performing a scientific task. This capability of the BBIC architecture, shared by most intelligent control concepts, supports the use of conventional controllers, when applicable, within highly complex systems. For formation flying applications, this capability is of critical importance due to the existence of legacy spacecraft control designs and the significant costs required to develop new designs. Interfacing intelligent controllers capable of coordinating the high-level actions of multiple spacecraft with existing low-level, conventional subsystem controllers responsible for implementing constituent actions is necessary to constrain design costs.

Hierarchically structured discrete event command and control methods also satisfy other fundamental requirements of formation flying systems, with behavior-based intelligent control methods being particularly applicable. For example, scalability is supported by the fact that the control architecture can be extended to multiple levels, thus supporting a hierarchy of vehicle clusters that represent assets in an overall distributed system. Scalability is also supported by the fact that intelligent controller designs can be generic and modular, thus incorporation of

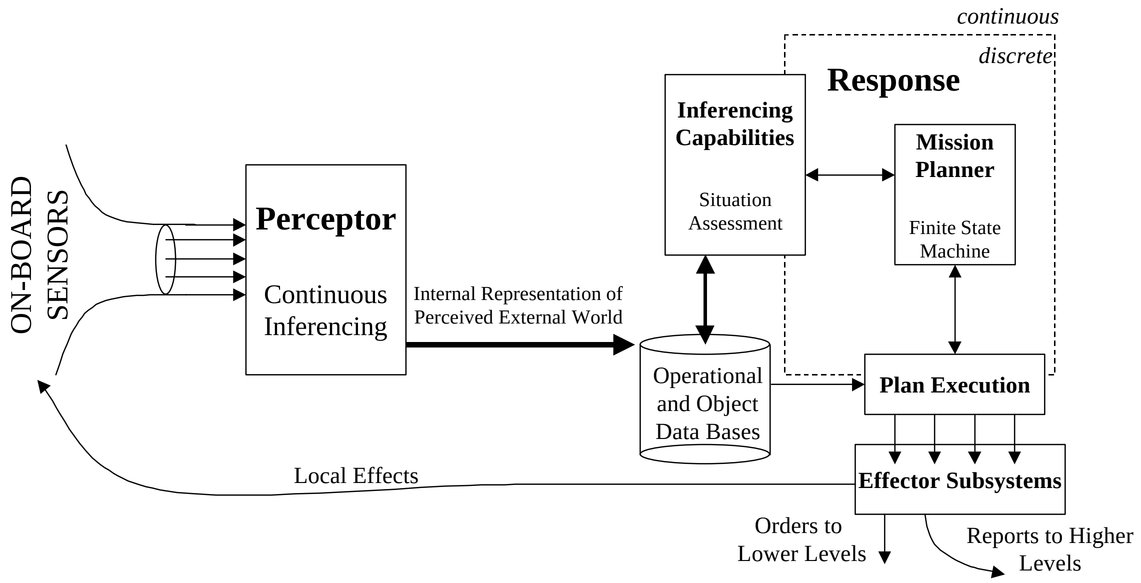


Figure 5. High Level Representation of the Behavior Based Intelligent Control Architecture

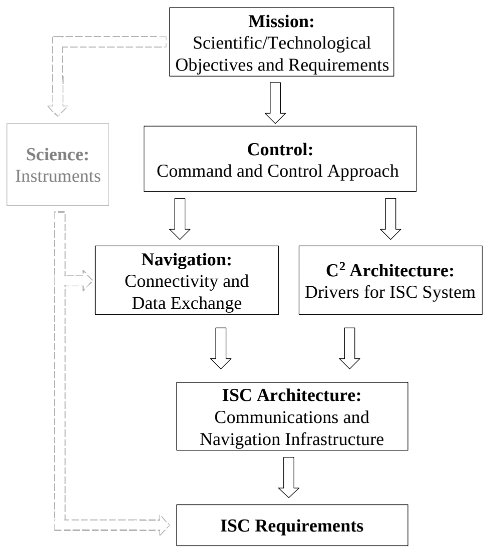
additional spacecraft is possible through replication and insertion rather than redesign.

At the same time the hierarchical structure serves to manage complexity by providing a structured decomposition of the system. Operational flexibility represents another fundamental requirement in formation flying applications supported by DES C^2 methods. This is realized by incorporating the ability to dynamically reconfigure the functional hierarchy among the distributed spacecraft, a necessity for systems in which supervisor loss or neutralization is possible. A simple example of this is the ability to autonomously promote a subordinate to a supervisory position in the absence of communication from the existing supervisor. In a broader sense adaptivity is facilitated by the autonomous generation of high level plans at both the component spacecraft level and at higher, supervisory levels. Ultimately coordination can be achieved by adequately assessing goal achievement and adapting system behavior to satisfy existing mission requirements.

Finally, efficiency, which is particularly crucial for individually limited small spacecraft, is achieved through the decomposition of behaviors and plans into constituent actions and tasks. By providing a formal mechanism for interaction among distributed spacecraft at the discrete event level, high-level plans can be exchanged with limited bandwidth resources and autonomously decomposed into complex behaviors and actions within the context of the environment of an individual spacecraft. The ability to operate with a high level of autonomy further reduces costly ground based control when multiple spacecraft comprise a system.

Implementation Issues

Implementation of a command and control strategy for formation flying relies upon autonomous interspacecraft communication among the distributed spacecraft. While communications among components in a distributed system is often assumed in the control literature, it is necessary to understand the way in which C^2 requirements drive the design of communications architectures, and how that connectivity in turn constrains system operations. For this research a distinction is made between the interspacecraft communications architecture and the functional control architecture. The former refers to the physical, interspacecraft communications infrastructure that can be modeled, for example, by the Open Systems Interconnect model;[5] the latter refers to the functional relationship among spacecraft in terms of system operations, such as C^2 or relative navigation. Both C^2 and relative navigation represent vital operations for formation flying due to the nature of the distributed spacecraft systems, which requires autonomous coordination and relative (and possibly absolute) position and velocity determination. Figure 6 illustrates the relationship between the development of functional control architecture and the interspacecraft communications architecture that supports it.



Mission:
Scientific/Technological Objectives and Requirements
Science: Instruments
Control:
Command and Control Approach
Navigation:
Connectivity and Data Exchange
C² Architecture:
Drivers for ISC System
ISC Architecture:
Communications and Navigation Infrastructure
ISC Requirements

Figure 6. Relationship Between Command and Control Architecture and Interspacecraft Communications (ISC) Architecture

Figure 6 illustrates the critical point that *the interspacecraft communications architecture of a distributed formation flying system is the infrastructure upon which system C^2 operations are executed*. For example, relative navigation will generally represent the dominant navigation computation for distributed sensing systems, which typically require knowledge of spacecraft baselines for correlating measurement data.

Obtaining relative position and velocity measurements may require direct communications links among spacecraft for the purposes of tracking crosslink signals, therefore the interspacecraft communications architecture must provide adequate connectivity among the distributed spacecraft. In general, C^2 operations will require dynamism in the functional control architecture that specifies the relationship among spacecraft because of the effects of stochastic failures that may occur during mission execution. Other control and science needs, such as ancillary data exchange for on-board data analysis in support of autonomous decisions, may produce additional constraints that affect the interspacecraft communications architecture.

In addition to the hierarchical DES C^2 structure presented above, the distributed nature formation flying systems may benefit from alternative functional control structures, such as centralized or fully distributed architectures as shown in Figure 7.

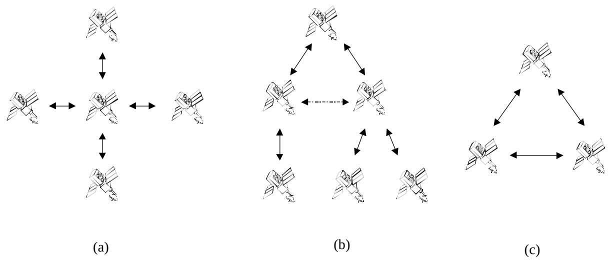


Figure 7. Canonical C^2 Architectures. (a) Centralized. (b) Hierarchically Distributed, with Possible Crosslinks. (c) Fully Distributed.

To support these architectures the interspacecraft communications architecture must provide the necessary connectivity.

For designs in which the functional control architecture and the interspacecraft communications architecture are identical, C^2 and relative navigation algorithms are directly supported by the communications infrastructure. However, particularly in systems in which new control methods are implemented upon existing communications architectures, additional constraints may develop that must be addressed by the distributed control algorithm. For example if communications use a star architecture in which subordinate spacecraft can only communicate through a supervisor (e.g., a "stovepipe") it is necessary to verify that latencies do not impact spacecraft coordination and that relative navigation requirements can be met.

Conclusions

The implementation of distributed spacecraft systems necessitates the development of intelligent command and control strategies applicable to the challenging space environment. These formation flying systems have been identified as critical to accomplishing campaign objectives that cannot be met by single spacecraft, and they therefore play a key role in future mission plans.

This research has presented a discrete event system-based approach to command and control that has been applied to highly complex, distributed autonomous systems. This approach is directly applicable to formation flying applications in general, and to microsatellite and nanosatellite distributed spacecraft missions in particular. As a hierarchically structured method of autonomous asset coordination, the DES C^2 approach presented facilitates the basic characteristics necessary to implement formation flying systems. This includes scalability, design flexibility, dynamic adaptation, and efficiency.

While these characteristics are relevant to formation flying systems in a general, they are fundamentally important in applications that involve individually limited spacecraft. By integrating the DES concept of hierarchical C^2 with a behavior-based system architecture, this approach to distributed spacecraft coordination directly supports efficiency by representing complex concepts in high-level behaviors. Planning and operations at the higher levels of the C^2 hierarchy occur at a level of abstraction sufficient to achieve mission objectives without fully representing all aspects of the system model. For example, interspacecraft communications can be utilized to exchange high level, discrete commands and control actions, thereby reducing crosslink traffic loads. At lower levels, behaviors can be autonomously decomposed into constituent actions and interfaces to conventional controllers and other effector subsystems can serve to implement detailed operations.

Significant research remains in this area of distributed, autonomous control. Of vital importance is the development of quantifiable, robust methods to handle complexity, both in the design and the operational phases of formation flying missions. This is particularly relevant to scalability issues, due to the stated desire by the space community to deploy systems involving tens and hundreds of small spacecraft to perform distributed sensing operations.

Acknowledgement: The author wishes to acknowledge the support of the NASA Goddard Space Flight Center's Guidance, Navigation and Control Technology Office for facilitating portions of this research.

References

1. Albus, J. S., "An Engineering Architecture for Intelligent Systems," Proc. of the AAAI, Fall Sy. Series, MIT, Cambridge, MA 1996.
2. Albus, J. S., "A Reference Model Architecture for Intelligent Systems Design," in P. J. Antsaklis and K. M. Passino, editors, "An Introduction to Intelligent and Autonomous Control," Kluwer Academic Pub., Boston 1993.
3. Antsaklis, P. J., and K. M. Passino, editors, "An Introduction to Intelligent and Autonomous Control," Kluwer Academic Pub., Boston 1993.
4. Bauer, F., J. Bristow, D. Folta, K. Hartman, D. Quinn, and J. P. How, "Satellite Formation Flying Using an Innovative Autonomous Control System (AutoCon) Environment," AIAA/AAS Astrodynamics Spec. Conf., New Orleans, LA 1997.
5. Black, U. D., "OSI: A Model for Computer Communications Standards," Prentice-Hall, Englewood Cliffs, NJ 1991.
6. Brooks, R. A., "A Robust Layered Control System for a Mobile Robot," IEEE J. of Robotics and Automation, Vol. 2, No. 1 1986.
7. Cassandras, C. G., "Discrete Event Systems: Modeling and Performance Analysis," Askin Assoc., Inc., 1993.
8. Das, A. and R. Cobb, "TechSat21 – Space Missions Using Collaborating Constellations of Satellites," 12th AIAA/USU Conf. on Small Satellites 1998.
9. Fordor, G. A., "Ontologically Controlled Autonomous Systems: Principles, Operations and Architectures," Kluwer Academic Pub., Boston 1998.
10. Heelis, R, study chair, "Geospace Multiprobes," Report from the Science Definition Team, NASA, December 1997.
11. How, J. P., R. Twiggs, D. Weidow, K. Hartman, and F. Bauer, "Orion: A Low-Cost Demonstration of Formation Flying in Space Using GPS," AIAA-98-4398, 1998.
12. Hsu, A., F. Eskafi, S. Sachs, and P. Varaiya, "The Design of Platoon Maneuver Protocols for IVHS," Program on Advanced Technology for the Highway Research Report, April 1991.
13. Kumar, R. and J. A. Stover, "A Behavior-Based Intelligent Control Archtecture with Appliction to Underwater Vehicles – Part I: Fundamentals," in process, 1998.
14. Kumar, R., J. A. Stover, and A. Kiraly, "A Behavior-Based Intelligent Control Archtecture with Appliction to Underwater Vehicles – Part II: Discrete Event Systems Modeling and Analysis," in process, 1998.
15. Kumar, R. and M A. Shayman, "Formulae Relating Controllability, Observability, and Co-Observability," Automatica, Vol. 34, No. 2, March 1998.
16. Kumar, R., V. K. Garg, and S. I. Marcus, "Language Stability and Language Stabilizability of Discrete Event Systems," SIAM J. Control and Optimization, Vol. 31, No. 5, September 1993.
17. Lima, P. U., and G. N. Saridis, "Design of Intelligent Control Systems Based on Hierarchical Stochastic Automata," Series in Intelligent Control and Intelligent Automation, Vol. 2, World Scientific Pub., New Jersey 1996.
18. Mauk, B. H., R. W. McEntire, R. A. Heelis, and R. F. Pfaff, Jr., "Magnetospheric Multiscale and Global Electrodynamics Missions," in Sun-Earth Connections, Geophysical Monograph 109, edited by J. L. Burch, American Geophysical Union, Washington, D. C. 1999.

19. Peluso, E., J. Goldstine, S. Phoha, et. al., "Hierarchical Supervision for the Command and Control of Interacting Automata," Proc. of the Sym. on Command and Control Research, Monterey, CA, June 1994.
20. Peluso, E. M. M., "A Hierarchical Structure of Interacting Automata for Modeling Battlefield Dynamics: Controllability and Formal Specification," Ph.D. Thesis, Dept. of Computer Science, The Pennsylvania State University 1996.
21. Phoha, S., J. Stover, and P. A. Stadter, "Ocean Sampling Mobile Network Controller," Sea Technology Vol. 38, No. 12, December, 1997.
22. Phoha, S., J. Stover, R. Gibson, E. Peluso, and P. A. Stadter, "Autonomous Ocean Sampling Mobile Network Controller," 10th Int'l. Sym. on Unmanned Untethered Submersible Tech., New Hampshire 1997.
23. Phoha, S., E. Peluso, P. A. Stadter, J. Stover, and R. Gibson, "A Mobile Distributed Network of Autonomous Undersea Vehicles," AUVSI '97 Proceedings, Baltimore, MD 1997.
24. Phoha, S., S. Sircar, A. Ray, and I. Mayk, "Discrete Event Control of Warfare Dynamics," Proc. 1992 Sym. on Command and Control Research, June 1992.
25. Ramadge, P. J., and W. M. Wonham, "Supervisory Control of a Class of Discrete Event Processes," SIAM J. of Control and Optimization, Vol. 25, No. 1, January 1987.
26. Raney, R. K. and R. F. Gasparovic, "POES Companion: Objective, Methodology, and Benefits," Acta Astronautica, Vol. 39, No. 9-12, 1996.
27. Saridis, G. N., "Architectures of Intelligent Controls," in M. M. Gupta and N. Sinha, editors, Intelligent Control Systems, IEEE Press, Piscataway, NJ 1995.
28. Stadter, P. A., W. S. Devereux, R. A. Denissen, et. al.m "Interspacecraft Communications Architectures for Formation flying," To appear in AIAA 1999 Space Tech. Conf. and Exp., 1999.
29. Stadter, P. A., W. S. Devereux, R. A. Denissen, et. al., "Interspacecraft Communications for Formation Flying," NASA/ESTO Study Report, Internal memorandum, NASA-GSFC, 1999.
30. Stadter, P. A., "A Neural Architecture for Fuzzy Classification with Applications," Ph.D. Thesis, Department of Electrical Engineering, The Pennsylvania State University, 1997.
31. Stadter, P. A. and A. K. Garga, "A Neural Architecture for Fuzzy Classification with Application to Complex System Tracking," Proc. 1997 Int'l Conf. on Neural Networks, Vol. IV, pp. 2439-2444, Houston, TX 1997.
32. Stover, J. A., D. L. Hall, and R. E. Gibson, "A Fuzzy-Logic Architecture for Autonomous Multisensor Data Fusion," IEEE Trans. Indust. Electronics, Vol. 43, No. 3, June 1996.
33. Wonham, W. M., "A Control Theory for Discrete-Event Systems," in M. J. Denham and A. J. Laub, editors, NATO ASI Series, Vol. F47, Advanced Computing Concepts and Techniques in Control Engineering, Springer-Verlag, Berlin 1988.