CLUSTER FLIGHT APPLICATION ON SYSTEM F6

Dr. Sun Hur-Diaz[(1)] **and Brendan O'Connor**[(2)]
Emergent Space Technologies, Inc.,
6411 Ivy Lane, Suite 303, Greenbelt, MD, 20770, USA, 301-345-1535
[(1)]*sun.hur-diaz@emergentspace.com*
[(2)]*brendan.oconnor@emergentspace.com*

Abstract: *The Cluster Flight Application (CFA) is a core application developed under the DARPA System F6 Program to provide guidance, navigation and control (GN&C) services to a cluster of spacecraft. System F6 seeks to "demonstrate the feasibility and benefits of fractionated space architectures, wherein the functionality traditionally co-resident within a single, large, 'monolithic' satellite is delivered by a cluster of wirelessly-interconnected modules [spacecraft] capable of sharing their resources and utilizing resources found elsewhere in the cluster" [1]. The CFA, when deployed on a spacecraft, would enable it to safely participate in the System F6 fractionated cluster by providing coordinated services that the spacecraft could not provide for themselves, such as relative navigation and cluster-coordinated maneuvers, to maintain the cluster safely and efficiently without extensive ground crew support.*

Keywords: *System F6, cluster flight, navigation, guidance, control, flight software.*

1. Background

System F6 (Future, Fast, Flexible, Fractionated, Free-Flying Spacecraft United by Information Exchange) is a DARPA program whose goal is to demonstrate the feasibility and benefits of disaggregated—or fractionated—space architectures, wherein the functionality traditionally co-resident within a single, large, "monolithic" satellite is delivered by a cluster of wirelessly-interconnected modules capable of sharing their resources and utilizing resources found elsewhere in the cluster. A module is defined, in this program, as an independent spacecraft that is part of a fractionated cluster and cluster network, and that makes one or more of its components available as shared resources on the cluster network. Such an architecture enhances the adaptability and survivability of space systems, while also shortening development timelines and reducing the barrier-to-entry for participation in the national security space industry [1, 2, 3, 4]. In this paper, the terms, module and spacecraft, are used interchangeably.

When the current System F6 program was started in 2011, the high-level objectives of the program were [1]:

1. Capability for long-duration maintenance of a cluster and cluster network, and to add and remove modules to/from the cluster and cluster network. A cluster is a set of modules with nearly equal orbital elements and able to maintain a stable relative geometry to support wireless communications.

2. Capability to securely share resources, such as a spacecraft component registered and made accessible across the cluster network for use by applications that may or may not be co-resident on the same physical spacecraft module, across the cluster network and among payloads or users in multiple security domains. A security domain is an enclave

within which data can be freely shared among users, between applications, and across resources, but beyond which data is controlled.

3. Capability to autonomously reconfigure the cluster to retain safety- and mission-critical functionality in the face of network degradation or component failures.

4. Capability to perform a defensive cluster scatter and re-gather maneuver to rapidly evade a debris-like threat; specifically, 5 minutes after a command is received from the ground, each module should be at least 10 km from where any module would have been under normal orbit operations (i.e., if no scatter maneuver had been initiated), and each pair of modules are as far apart as possible; the planning and execution of the scatter and re-gather maneuvers shall be performed without intervention or communication from ground operations.

One of the key artifacts of System F6 is the F6 Developer's Kit (FDK), which is a set of open source interface standards, protocols, software, behaviors, and reference implementations thereof, necessary for an independent 3rd party, without a contractual relationship with or assistance from any System F6 performer, to develop a clean-sheet module design that can fully participate in a fractionated cluster. The FDK is composed primarily of three technical areas: Cluster Flight Application, Information Architecture Platform (IAP) and Wireless Inter-module Communications (WIC). The specific objectives of each of these technical areas are given in DARPA-BAA-11-01 [1].

Emergent Space Technologies, Inc. (Emergent) was selected to develop the Cluster Flight Application (CFA) – the algorithms and flight software (FSW) - that enable multi-body cluster flight safe from collisions, efficient relative navigation, relative station-keeping, and cluster reconfiguration—including rapid defensive maneuvering.

The demonstration mission of the key functional attributes of fractionated architectures constituting these high-level objectives was subsequently de-scoped from the program along with objectives such as the defensive scatter and re-gather. However, the development of CFA continued and is planned to be completed in June 2014.

This paper provides an overview of CFA including the primary use cases driving its design and development. It describes the CFA software architecture and its services followed by a summary of software development and verification and validation (V&V) methodology.

2. CFA in a Nutshell

The Cluster Flight Application (CFA) is a distributed flight software package that provides guidance, navigation, and control (GN&C) services that enable multiple modules to safely participate in mission applications that require them to orbit in close proximity to each other. The term cluster as used here means a set of spacecraft orbiting together, whether they are flying in formation or merely in proximity. CFA can support either concept.

CFA provides services that the individual spacecraft cannot provide for themselves, such as relative navigation and cluster-coordinated maneuver planning and execution, to maintain the space-

craft orbits safely and efficiently. CFA is designed to run on-board the spacecraft as flight software without much ground intervention. As such, it is designed to complement and integrate with the existing spacecraft Command and Data Handling (C&DH), ground links, and other components of the mission technology suite. This includes GPS and relative navigation sensors, shared payloads, data-link layers, etc. to achieve the mission goals using shared resources. It also supports dynamic resource allocation, fault tolerance, load balancing, failover, and middleware-based information exchange.

One of the underlying goals of CFA is to provide a framework whereby future enhancements to CFA can be easily injected. To this end, the major CFA functionalities are designed as services that can reside on any processing node on any module that has sufficient computational resource and can communicate with the rest of the cluster. The processing node can even be on the ground segment that is connected to the cluster network via a networked link. These services communicate with each other via a messaging bus provided by the computing infrastructure. CFA services are flexibly deployable and can move from one networked computing node to the next as needed to best utilize available resources. This approach works well in a range of scalable mission architectures from a single module mission up to 20 modules in a cluster or formation.

CFA, when deployed on a module, enables it to safely participate in the System F6 fractionated cluster by providing coordinated services that the modules could not provide for themselves, such as relative navigation and cluster-coordinated maneuvers, to maintain the cluster safely and efficiently without extensive ground crew support. Key System F6 objectives to be met by the CFA include semi-autonomous long-term cluster maintenance, ingress of a module into the cluster, egress of a module out of the cluster, passive safety, autonomous scatter and re-gather of the cluster, and reconfiguration of the cluster geometry. Semi-autonomous, in this paper, means requiring a minimal ground crew to operate a cluster where the size of the ground crew is invariant with the number of modules in the cluster.

Key CFA attributes are:

- *Developed and Tested as Flight Software* – CFA was developed as flight software and has been designed, developed, documented and tested with this in mind.
- *Service-based Architecture* – CFA was designed as a set of services communicating through a message bus. CFA software is agnostic regarding the operating system, hardware and messaging layer it runs on. This approach makes the software readily portable to a variety of common spacecraft platforms including VxWorks, core Flight Executive/Core Flight Software (cFE/CFS), OpenDDS/Linux and others.
- *State-based Spacecraft Management* - CFA provides management services to the spacecraft on its message bus network based on internal states. The ground controllers command the spacecraft into the states, and CFA provides the appropriate level and type of services based on the state. Ground operators control the services provided by CFA by setting CFA states of the spacecraft. See Section 4 for more details on the CFA states.
- *Flexible, Enhanced Navigation* – The navigation service provided by CFA is filtered state estimation and can incorporate GPS and additional navigation radiometric sensor information. In the event of a GPS failure or outage, CFA navigation can continue to provide

navigation information using additional sensors (such as range derived from radio cross-links).

- *Universal Maneuver Planning* – Coordinated maneuver planning is provided for up to 20 satellites flying in a formation or cluster. Maneuvers take into account the specific performance characteristics of each spacecraft and mission constraints such as cross-link radio range and minimum approach distances.

Figure 1 summarizes the key features of CFA.

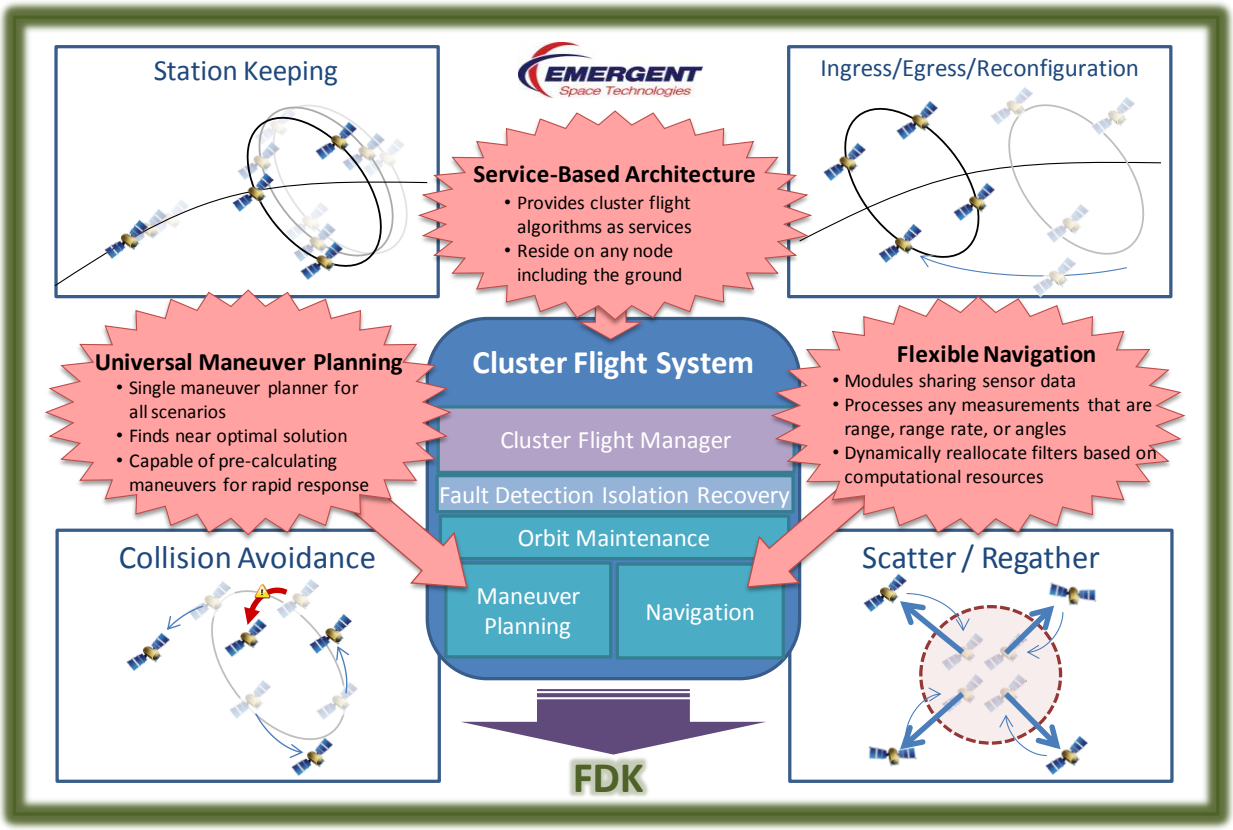

Figure 1. Summary of Cluster Flight Application

3. CFA Use Cases

The high-level program objectives provided by DARPA were analyzed, and a set of use cases that drive the CFA software requirements was developed. Areas of functionality were broken out into nominal and off-nominal categories in the use case model which define the required functionality and high-level architecture. For each use case category, multiple use cases were generated to further define behavior including basic, alternate, and exception flows, as applicable. Software was developed using agile software development processes where CFA functionality associated with the use cases was incrementally added and tested through many 3-week iteration development cycles or sprints. Figure 2 shows how the System F6 objectives flow down to the CFA system-level functional and performance requirements and to the use case specification documents and activity diagrams which are captured in the Enterprise Architect (EA) software package. The use cases flow down to the design of the CFA software components and the development of the system test scenarios.

Analysis of the CFA requirements included the development of the concept of operations for a typical cluster flight mission. Various cluster design parameters, such as cluster size, number of modules in the cluster, navigation noise, altitude of the cluster orbit and cluster configuration, were also studied through a parametric performance model (PPM) to determine their effects on station-keeping delta-V, probability of collision, and relative orbital geometry. An innovative dispersion analysis tool based on nonlinear propagation of covariance matrices was developed to perform this study [5]. Key considerations in the cluster design are summarized by Schmidt [6].

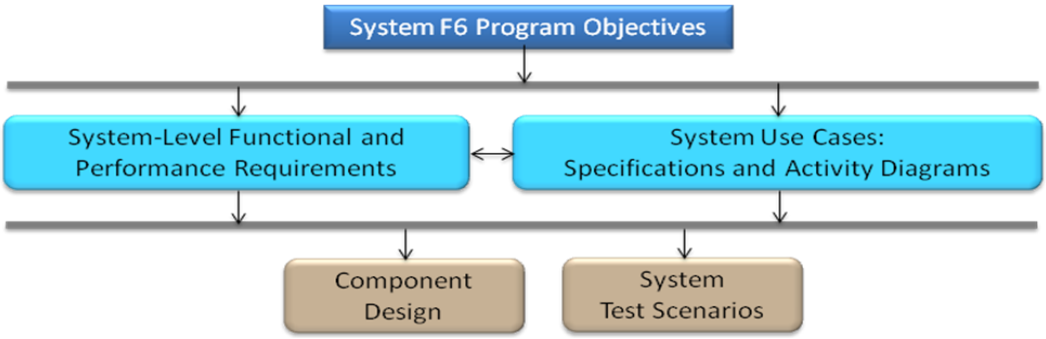

Figure 2. CFA Software Development Process

The primary set of use cases supported by CFA is listed in Table 1.

Table 1. Cluster Flight Application Use Cases

Initialize and Check-out Cluster Flight
Monitor CFA Component Status from the Ground
Start/Re-Start CFA
Store and Apply CFA Component Configuration Data
Nominal Operations
Navigation with GPS Only
Start or Terminate Relative Range Backup
Closed-loop Cluster Monitoring (including collision monitoring)
Generate a Maneuver Plan
Distribute a Maneuver Plan
Execute a Module Maneuver Plan
Inspect CFA Environment Data
Report CFA Environment Data Status
Store and Apply CFA Environment Data
Ground Updates Module Data
Ground Adds a Module to Cluster Flight Application
Ground Removes a Module from Cluster Flight Application
Ground Sets Cluster Station-keeping Mode
Inspect CFA Common Module Data
Report CFA Common Module Data Status
Store and Apply CFA Common Module Data
Change the Relative Orbit of a Module
Change the State of a Module
Off-Nominal Operations
GPS Outage Without Backup Navigation Source
GPS Outage with Backup Relative Range Based Navigation Source
Relative Range Trajectory Degradation without Backup Navigation Source

The navigation use cases assumes the use of Global Positioning System (GPS) sensors and the possible presence of relative range measurements between modules, e.g., from the WIC system. Not all use cases are equal in complexity. Some are relatively trivial such as Monitor CFA Component Status from the Ground which involves establishment of telemetry data for each CFA component, while others, such as Close-loop Cluster Monitoring, require efficient algorithm development, detailed systems engineering analysis, comprehensive trade studies, and thorough systems testing. Some use cases, such as Change the Relative Orbit of a Module, was the result of multiple iterations of systems engineering analysis and design steps leading to one simple implementation for all three of the System F6 objectives: ingress, egress, and reconfiguration of a module.

Many more use cases than those listed in Table 1 were analyzed and developed before the demo mission de-scope. In particular, the scatter and re-gather use cases were analyzed in detail and corresponding prototype software capability was developed and demonstrated through high-fidelity simulations. The design, algorithms, and results of scatter and re-gather are provided in detail by Brown [7] and Duffy [8]. With the de-scope of the demo mission, work on scatter and re-gather was suspended; however, this capability can be included in the CFA software for future applications, if needed.

4. CFA Service-Based Architecture

The service-based CFA architecture takes advantage of middleware-based Information Architecture. Figure 3 shows an illustration of a service-based architecture where cluster flight services are provided from a virtual cluster flight system. A service-based architecture has many benefits. It is evolvable, so new services, clients, transactions can be easily added. It is adaptable in that the services can be moved from processing node to node provided that communication between the nodes is available. The nodes may reside on separate modules in space, or even on the ground, if the latency is acceptable, where high performance processing capability may be available. This service-based architecture is robust, able to survive network disruptions and node failures [9].

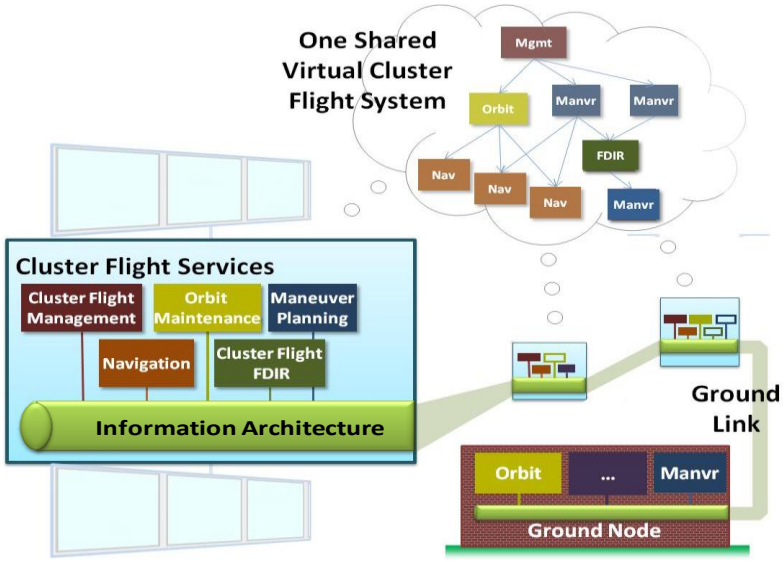

Figure 3. Service-Based CFA Architecture

The concept of a service-based CFA, distributed over several heterogeneous nodes, is illustrated in Figure 4, which shows a sample cluster consisting of four modules (or spacecraft). The GN&C functions of relative navigation, flight management, collision avoidance, station-keeping, and module maneuvering come from services that run on the individual spacecraft or on multiple spacecraft. The Message Bus shown on each spacecraft in the figure extends across the inter-spacecraft radio links and provides a common infrastructure and data connectivity between the service instances. Since a persistent, high-data rate connection to the ground was part of the System F6 infrastructure, the Message Bus is extended to the ground nodes. In this example, the Navigation service is deployed on every spacecraft, while the Maneuver Planning, Orbit Maintenance, and Cluster Flight Management services are hosted on Spacecraft 1 which supports the processing requirements of these services. The CFA services are described in more detail in the next section.

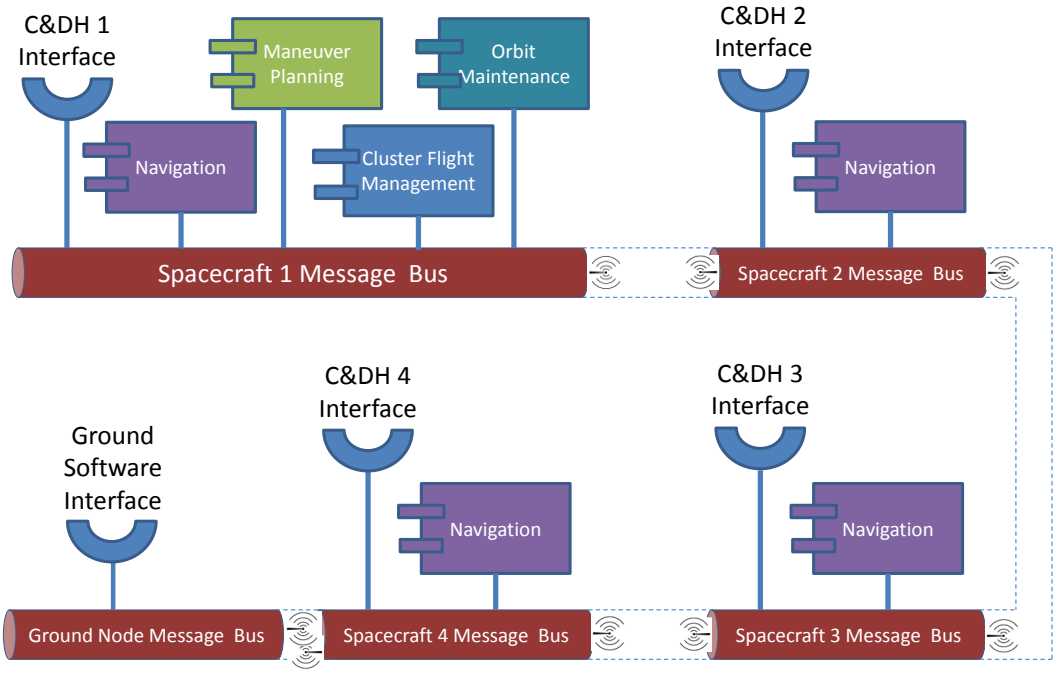

Figure 4. Sample Deployment of Cluster Flight Services

Client spacecraft that need to make use of CFA's services connect via the Messaging Bus and receive services through messages. Services provided by CFA are available to any module or payload. A payload application, for example could subscribe to the navigation solution for the spacecraft it was hosted on and receive navigation information directly from CFA.

CFA nominally functions without intervention from the ground. Ground operators are required for planning and commanding cluster orbit changes and monitoring the function of CFA only. The navigation services provided by CFA are meant to supplement the spacecraft's own navigation for safe cluster control. Maneuvers are provided by CFA at an abstract level as maneuver times and burn parameters that can be translated by the individual spacecraft into maneuver commands specific to that vehicle.

Since CFA is a service-based application, it requires at least a message-oriented middleware to run, but it is designed to be platform agnostic and independent of the underlying computing infrastructure in a general manner so that it could run on a number of platforms. CFA has already been ported to a number of operating system and messaging platforms throughout its development. The architecture is designed to spread out the computational load and minimize the required inter-spacecraft communications bandwidth. The individual services have defined interfaces and usage patterns, so they can be tested by themselves and put together to form a system with a higher function. The individual services can be upgraded and replaced over time. CFA is built to be a robust, distributed system that adapts and evolves over time.

4.1 CFA Services

The component services included with CFA include the Cluster Flight Manager (CFM), Orbit Maintenance Service (OMS), Navigation Service (NAV), Maneuver Planning Service (MPS), and Maneuver Monitoring Service (MMS). The component services work together to enable the cluster flight capabilities provided by CFA. Figure 5 shows the interaction between the various CFA services. CFM serves as the main hub for passing data and messages between the other services. An instance of the NAV service ideally is installed on each module of the cluster. Each instance of NAV processes Global Positioning System (GPS) data and relative range measurements for a single spacecraft through an extended Kalman filter to generate real-time state and covariance updates. These are provided to CFM, which maintains an inventory of the module states and characteristics. OMS hosts the guidance algorithms for monitoring the cluster states and probability of collision and determining whether maneuver planning is required. If so, OMS interacts with MPS to generate a set of maneuvers and provides those back to CFM. CFM then distributes the maneuvers to each module's MMS for processing and execution using the onboard propulsion system. With this design architecture, CFA is capable of providing its services to the cluster while sharing resources across the cluster modules and the ground.

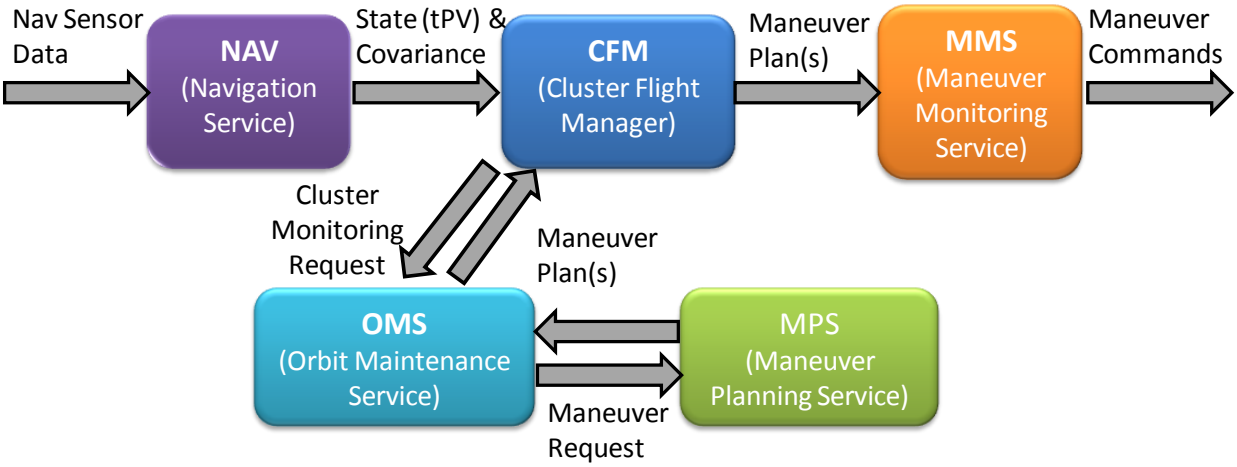

Figure 5. Interactions of the CFA components

Cluster Flight Manager (CFM)

CFM is responsible for the automation and coordination of all CFA services, maintaining an inventory of all the modules that are members of the cluster, and managing the states of the cluster and of each module. Other key functionalities of CFM include scheduling of monitoring the cluster collision probability and management of maneuver distribution to the spacecraft in the cluster.

CFM manages services to the modules based on their internal states. The ground controllers command the module into a state, and CFA provides the appropriate level and type of services based on that state. A module can have one of four states which are summarized below:

- **Planned** - Modules in the Planned state have been added to the cluster inventory but may or may not be in orbit yet. Modules in this state are not provided station-keeping, collision monitoring, or navigation services.
- **OutOfCluster** - Modules in the OutOfCluster state have been added to the cluster inventory and are in orbits that may or may not be within radio cross-link range but are under the direct control of the ground or are uncontrollable. Modules in this state are not provided station-keeping or collision monitoring services but may be provided navigation services.
- **Holding/InCluster** - Modules in the Holding/InCluster state have been added to the cluster inventory and are within cross-link radio range. Modules in this state are under the control of CFA which is providing station-keeping, collision monitoring, and navigation services.
- **Reconfiguring** - Modules in the Reconfiguring state have been added to the cluster inventory and are transitioning from one orbit to another orbit. Modules in the Reconfiguring state are under the control of CFA which is providing closed-loop maneuver planning, collision monitoring, and navigation services. A module that is ingressing or egressing would be in this state.

Transition between states can be realized by ground command. Some are performed autonomously when transitioning through the Reconfiguring state or when an error occurs. The Holding/InCluster state is typically used during the operational phases of a parking orbit and the nominal mission orbit.

Maneuver Monitoring Service (MMS)

MMS is a service present on every module. It receives module maneuver plans from CFM and stores them until it is time to send the time-tagged maneuver message to the bus through a module proxy which translates the maneuver message into commands that are processed by the bus C&DH subsystem.

Orbit Maintenance Service (OMS)

OMS implements the guidance and station-keeping strategies during the various operational scenarios required by the cluster or formation. Table 2 summarizes the main functions of OMS. It

monitors the module position geometry parameters and state information and provides maneuver targets for use in maneuver planning. For example, for cluster orbit maintenance, it specifies the target relative orbit elements that are desired, the maneuver window, and the constraints associated with the spacecraft such as minimum and maximum delta-V.

Table 2. Orbit Maintenance Service (OMS) Functions

Maneuver Request Management	• Manages maneuver requests and results • Triggers request timeout
Station Keeping	• Maintains cluster (and any formation requirements) • Minimizes the risk of collision
Closed-Loop Maneuver Planning	• Checks maneuver plan performance • Triggers maneuver replanning, if necessary
Collision Probability	• Monitors collision probability
Scatter Preplanning	• Propagates module states • Creates preplan requests

OMS continuously monitors the relative state of all of the modules by periodically propagating their orbit state including planned maneuvers to predict collisions and determine when they will leave their control boxes. OMS would also be responsible for managing the pre-calculation of maneuvers required for rapid response scenarios like scatter. The station-keeping strategy that has been implemented is discussed in detail by Ruschmann [10]. The details of the scatter strategy are provided by Brown [7] and Duffy [8].

The onboard computation of the cluster probability of collision is based on a new method developed for System F6 that is based on weighted log average of Mahalanobis distance and maximum instantaneous probability of collision. The weights were selected by applying least squares to the results from a large range of scenarios and Monte Carlo simulations used to generate "truth" data for natural motion circumnavigation orbits, co-elliptic overhead flyby orbits, and string-of-pearls station-keeping orbits [11].

Maneuver Planning Service (MPS)

MPS processes requests for specific coordinated cluster maneuver plans as shown in Figure 6. It takes as inputs maneuver plan goals, design parameters, and constraints (such as initial module states, final target states, maneuver window, and propulsion limits) and generates multi-burn maneuver plans. Its flexible algorithm is designed to universally compute the delta-V maneuvers for all the orbit control scenarios required by System F6. Through a robust heuristic search method based on simulated annealing, MPS finds the optimal maneuver times and targets that satisfy the cluster control objectives and constraints. An underlying efficient multi-burn solver based on linear programming generates the optimum delta-V plans for a given set of initial and final conditions selected by the heuristic search. The details of the simulated annealing and the multi-burn solver are given by Brown [7].

Figure 6. Maneuver Planning Service

Navigation Service (NAV)

NAV estimates cluster module states based on data from spacecraft sensors and range data derived from the radio cross-links. It provides high-quality absolute and relative navigation solutions for modules that they may not be able to generate themselves. NAV provides position and velocity information for all the modules in the cluster. Because of the criticality of the navigation information, robustness and reliability are the main driving factors of the NAV architecture. Although the NAV framework is built to accept a variety of measurement types, including range, range rate, and angles, its nominal mode of operation uses GPS pseudo-range measurements. The NAV filter solutions based on these GPS measurements provide sufficiently accurate relative navigation for efficient cluster control for System F6. The NAV architecture is designed to provide robustness to failures in either the spacecraft cross-links or in the GPS information through its fault detection, isolation, and recovery function.

The distributed navigation architecture consists of two filters in each module (a GPS filter and a relative range filter) in addition to a navigation manager that selects which filter result to publish based on a number of performance metrics such as normed innovation squared and other measurement rejection criteria. This fault detection, isolation, and recovery architecture can maintain accurate relative navigation even if there are interruptions to GPS signals or failures to key GPS receiver components and is an innovation that is a significant improvement on the state-of-the art. It is particularly beneficial to clusters with more than two modules and to small satellites that cannot handle the additional weight for backup GPS receiver components. More details of the architecture, design, algorithms, and performance of the Navigation Service are provided by Schmidt [12].

5. Prototype CFA Software Development

For prototype software development, Emergent leveraged an emerging MATLAB capability to auto-code MATLAB functions into C and C++ [13]. This approach to software development enabled algorithm designers to rapidly implement prototype cluster flight algorithms and leverage MATLAB plotting and data evaluation tools. The MATLAB algorithms are peer-reviewed by the Principal Investigator, Software Architect, and CFA software leads for design and functionality. In addition, a static code analysis tool is used to evaluate the resulting auto-coded C and the human coded C++ code. This approach has shown to also have benefits later in the flight software validation phase, as developers can go back to the original prototyping arena to address and fix algorithm issues without major downstream implications as long as the external component interfaces are maintained. The core functionality for most applications consists of some mix of

auto-coded C-code and hand-written C++ classes as shown in Figure 7. CFA services that use auto-generated code libraries include MPS, OMS, and NAV.

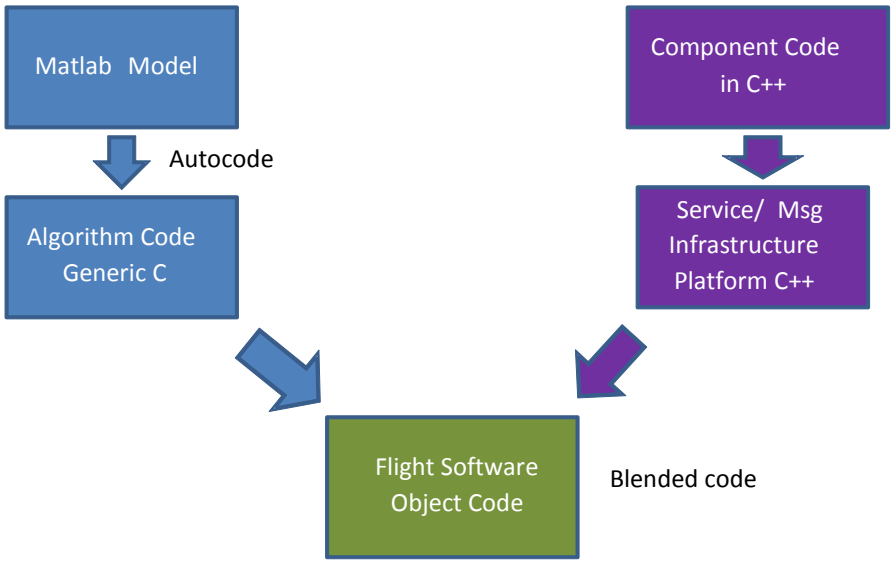

Figure 7. CFA Code Development Flow

During development and eventual deployment, CFA services were always expected to be hosted and tested in different environments. A reusable core software platform-independent implementation is coded in C++ and defines the service interface functions and data types. This core is isolated behind a message bus adaptor and several supporting interfaces such as logging, file input/output, and system time. The message bus adaptor and host application meet the deployment environment needs without requiring changes to the private implementation behind the adaptor.

6. Implementation on the core Flight Executive (cFE) Platform

CFA was originally planned to be integrated with the custom F6 operating system (F6OS) that was being developed by the Information Architecture Platform (IAP) technical area. When the demonstration mission was de-scoped from the program, the development schedule of CFA and System F6 IAP diverged, and the integration of CFA with F6OS was not deemed practical. In place of System F6 IAP, the core Flight Executive (cFE) and the Core Flight Software (CFS) were selected to continue the development of CFA. cFE/CFS is a NASA Goddard Space Flight Center (GSFC) product that has significant flight heritage [14]. Rationale for choosing cFE include:

- A replacement for the System F6 IAP Services was needed to support:
  - Messaging
  - Layered Architecture
  - Partitions (ARINC 653)
  - Components
  - Startup/shutdown monitoring
  - Inter-spacecraft Communication
- cFE is open source and widely used

- cFE provides basic services for FSW
- cFE provides a known FSW environment with broad support at NASA and interest from DoD
- cFE provides maximum flexibility for future missions
- CFS provides additional services needed by many spacecraft
- Porting to cFE inexpensively moves the CFA towards flight-ready status
- Provides a spacecraft-like environment to ease the transition to a mission

7. Verification and Validation

The functionalities of the CFA are being verified using MATLAB simulations and a high-fidelity embedded FSW simulation. A development environment called the Distributed Integrated Environment for CFA Analysis, Simulation, and Testing (DIECAST) was developed that enables the CFA FSW to be exercised in a range of on-orbit scenarios with a variable number of spacecraft, spacecraft properties, orbit elements, and relative cluster geometry. Spacecraft sensor and effector models were developed that enables emulation of inter-module communication within a cluster and coordinated flight control scenarios. Sensor data and control commands are exchanged between the multi-spacecraft simulation and the CFA FSW. The resulting navigation and trajectory performance data is logged by DIECAST and used for performance assessment throughout the FSW development lifecycle. DIECAST enables developers to run the CFA FSW in a desktop environment that provides a reasonable facsimile of the flight environment with a high-fidelity simulation that can be run faster than real-time [15].

DIECAST is built upon the NASA Trick Simulation Environment (Trick), which provides a framework for building model-based simulations [16, 17]. Trick includes an executive scheduler that allows the models and CFA FSW to be executed at specified rates throughout the simulation. Scenarios are configurable at run-time, with the intent to allow a single build of the simulation to support testing with a configurable number of spacecraft with unique geometry and initial conditions, beginning at any phase of flight. It supports the distributed nature of the CFA FSW execution among the members of the cluster even with the added complexity of changes in the deployment of the FSW components and spacecraft interfaces. The DIECAST simulation simplifies the problem of FSW verification by providing a common FSW interface for embedded and hardware-in-the-loop (HWIL) testing, and allowing all scenarios to be evaluated with run-time inputs to a single build of the simulation.

Verification and validation test scenarios were developed based on the CFA system-level use cases to exercise as many flows as possible through the software functionality. For each test scenario, a unique input file was created to exercise the functionality of the use case flows, including environment set-up, ground commands to CFA, and logging data to verify the system responses. As CFA builds were released for testing, tags were created to freeze the CFA FSW and DIECAST build environments, to ensure stability and repeatability of behavior. Test cases were executed against the tags and the recorded data was analyzed to determine the CFA system behavior. Any discrepancies in behavior between the use cases and the test results were entered as defects in the defect tracking tool. Following testing cycle completion, all test inputs files, data setup, and results were committed to configuration managed repositories for regression testing and comparison against future version releases of the software. To track results of each testing cycle, a Verification Traceability Matrix (VTM) was created to map use cases to test case rela-

tionships and indicate PASS/FAIL status. Several steps are taken to ensure quality CFA flight software.

- Solid Bi-directional Requirements/Use Case/Test Traceability
- Peer Reviews of Documentation, Products
- Static Analysis of Code
- Peer Code Reviews
- Unit Testing
- Independent System Level Testing
- Integrated Narrative Testing
- Benchmarking Memory Footprint
- Component Performance Testing
- Defect Tracking/Resolution

8. Summary

A high-level summary description of Cluster Flight Application (CFA) flight software developed for System F6 was presented. The service based architecture of CFA consists of five core components: Cluster Flight Manager, Maneuver Monitoring Service, Orbit Maintenance Service, Maneuver Planning Service, and Navigation Service. CFA was developed for wide applications of cluster or formation flying missions. With well defined interfaces associated with each of the services, they are essentially applications by themselves that can be individually integrated with other mission flight software to support specific mission needs through a messaging bus. The architecture also allows for another application with similar functionality to interface with the CFA services. Furthermore, each service developed for System F6 was architected with maximum flexibility such that other algorithms can replace the reference implementation that is currently in each service. CFA is part of the F6 Developer's Kit that includes the software, its description, and the verification and validation results.

9. Acknowledgements

The authors would like to recognize the dedicated team of Emergent engineers who worked tirelessly in developing the CFA flight software and the underlying component applications through their innovation, ingenuity, and expertise. The authors would also like to thank DARPA for funding this work, NASA JSC for their support on Trick, NASA GSFC for their support on cFE/CFS, and Mathworks for their Pilot Support Program.

10. References

[1] DARPA, Broad Agency Announcement, System F6, Tactical Technology Office (TTO), DARPA-BAA-11-01, 20 Oct 2010.

[2] Brown, O. and Eremenko, P. "The Value Proposition for Fractionated Space Architectures." AIAA Space 2006. San Jose, CA: American Institute of Aeronautics & Astronautics. Paper No. AIAA–2006–7506.

[3] Brown, O. and Eremenko, P. “Fractionated Space Architectures: A Vision for Responsive Space.” 4th Responsive Space Conference. Los Angeles, CA: American Institute of Aeronautics & Astronautics. Paper No. AIAA–RS4–2006–1002.

[4] Brown, O., Eremenko, P., and Bille, M. “Fractionated Space Architectures: Tracing the Path to Reality.” Small Satellite Conference, Logan, Utah, 2009.

[5] Hur-Diaz, S., et al, "Computing Collision Probability Using Linear Covariance and Unscented Transforms," AIAA Guidance, Navigation, and Control and Co-located Conferences and AIAA Infotech@Aerospace 2013, Boston, MA, 19-22 Aug 2013.

[6] Schmidt, J., Phillips, M., Ruschmann, M., “Satellite Cluster Flight Design Considerations,” 24th International Symposium on Space Flight Dynamics, Laurel, Maryland, May 2014.

[7] Brown, A., Ruschmann, M., Duffy, B., Ward, L., Hur-Diaz, S., Ferguson, E., and Stewart, S., “Simulated Annealing Maneuver Planner for Cluster Flight,” 24th International Symposium on Space Flight Dynamics, Laurel, MD, April 2014.

[8] Duffy, B., Brown, A. G., Ruschmann, M. C., Ward, L. “Scatter Strategies for Cluster Flight.” 24th International Symposium on Space Flight Dynamics. Laurel, MD, April 2014.

[9] O’Connor, B., Brown A., Gordon K., Schmidt J., de la Torre R., “A Service-based Architecture for Automated Guidance, Navigation and Control Flight Software." 2013 Workshop on Spacecraft Flight Software. Pasadena CA, 10-12 Dec 2013.

[10] Ruschmann, M. C., Duffy, B., de la Torre, R., and Hur-Diaz, S. “Efficient Station-Keeping For Cluster Flight.” 24th International Symposium on Space Flight Dynamics. Laurel, MD, April 2014.

[11] Phillips, M., Hur-Diaz, S., "On-board Estimation of Collision Probability for Cluster Flight," Paper ASS 13-357 presented at the AAS/AIAA Space Flight Mechanics Meeting, Lihue, HI, February 10-14, 2013.

[12] Schmidt, J. and M. Phillips, “A Distributed, Redundant Navigation and Fault Detection System for DARPA System F6." AIAA Guidance, Navigation, and Control and Co-located Conferences and AIAA Infotech@Aerospace 2013. Boston, MA, 19-22 Aug 2013.

[13] MATLAB Coder. The Mathworks Inc. <www.mathworks.com/products/matlab-coder/>

[14] Core Flight Executive (cFE). Code.NASA. < code.nasa.gov/project/core-flight-executive-cfe>

[15] Stewart, S. M., Ward, L., and Strand, S. "Distributed GN&C Flight Software Simulation for Spacecraft Cluster Flight.” AAS Guidance Navigation and Control Conference, Breckenridge, CO, Feb. 2014.

[16] Paddock, E., Lin A., Vetter K., Crues E., “Trick®: A Simulation Development Toolkit." AIAA Modeling and Simulation Technologies Conference and Exhibit. Austin TX, 11-14 Aug 2003.

[17] Innovative Simulation Toolkit for Constructing Simulations. NASA Technology Transfer and Commercialization Office. <http://www.nasa.gov/centers/johnson/techtransfer/technology/MSC-24492-1-sim-toolkit.html>