2015

Fusion: A Framework for Human Interaction with Flexible-Adaptive Automation Across Multiple Unmanned Systems

Allen J. Rowe

Sarah E. Spriggs

Daylond J. Hooper

FUSION: A FRAMEWORK FOR HUMAN INTERACTION WITH FLEXIBLE-ADAPTIVE AUTOMATION ACROSS MULTIPLE UNMANNED SYSTEMS

Allen J. Rowe, Sarah E. Spriggs
Air Force Research Laboratory
Dayton, Ohio

Daylond J. Hooper
Infoscitex
Dayton, Ohio

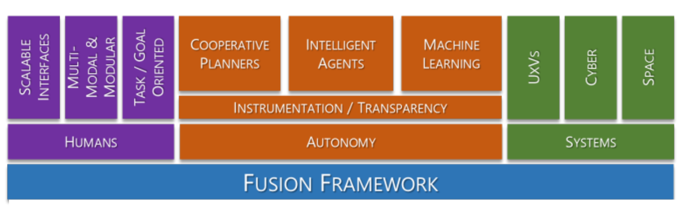
Future unmanned systems operators and heterogeneous unmanned systems must be able to work as agile synchronous teams to complete tactical reconnaissance, surveillance, and target acquisition related missions requiring the use of automation to assist the human operator. Interface research in this area is critical to the success of human-automation teaming, thus requiring a research test bed that brings together humans, autonomy, and systems. Fusion is a framework that enables natural human interaction with flexible and adaptive automation via the use of intelligent agents reasoning among disparate domain knowledge sources, machine learning providing monitoring services and intelligent aids to the operator, cooperative planners and advanced simulation through an instrumented, goal oriented operator interface that empowers scientific experimentation and technology advancement across multiple systems. There are four primary research threads that the Fusion framework is addressing to accomplish these goals: cloud-based simulation architecture; software extensibility; interface instrumentation; and, human-autonomy dialog through retrospection.

Autonomous systems are becoming an increasingly critical aspect of military operations. To enable these technologies, ever more capable autonomy frameworks are required, providing task-based management of a multitude of heterogeneous unmanned systems (UxSs) (USAF Chief Scientist, 2013). These technologies are still in their infancy, requiring extensive research to target increased understanding of how human operators can effectively coordinate and communicate with autonomous systems, among other issues in this domain. Robust autonomy-based frameworks enable evaluation of cooperation and coordination among widely disparate platforms such as remotely piloted unmanned systems (RPAs), autonomous unmanned systems, and adversarial components for ground, air, cyber, space, maritime, and submarine entities. Tying these interactions into an immersive user interface will improve evaluation of user behaviors and confidence in a low-risk environment. However, a unique challenge exists in unification of user interactions, autonomous platforms, and intelligent aids. A common drive is to push towards more autonomy, diminishing the user's involvement. Users can provide useful information to autonomous systems, and autonomy can be used to augment user capabilities, so an alternative is to develop and support symbiosis between users and systems. This symbiosis can be realized via a robust framework that provides user-tunable accessibility into this autonomy. This enables evaluation of user comfort, trust, and confidence with autonomous components. The associated ability to tune autonomy also drives future requirements for user interface design and accessibility.

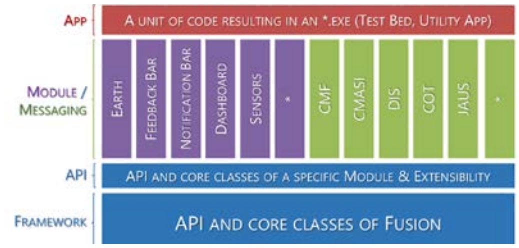
Fusion Overview

Fusion Framework

Fusion is a framework that enables natural human interaction with flexible and adaptive automation. It employs multiple components: intelligent agents, reasoning among disparate domain knowledge sources (Douglass, 2013); machine learning, providing monitoring services and intelligent aids to the operator (Vernacsics, 2013); cooperative planners (Kingston, 2009); and advanced simulation via an instrumented, goal-oriented operator interface (Miller, 2012). These empower scientific experimentation and technology advancement across multiple systems (see Figure 1). The Fusion Framework consists of a layered architecture supporting disparate research projects with a development kit to explore a variety



of research goals. The framework consists of four fundamental layers: (a) the core framework layer, (b) the extensibility and API layer, (c) the module / messaging layer, and (d) the application layer (see Figure 2). The core framework layer provides foundational software classes and an application programming interface (API). This layer enables functionality for module lifecycle, user profile, and display layout management. Additional features of this layer include system level notifications, multi-modal interactions and feedback, workspace management, asset management (vehicles, tracks, sensors, named areas of interest, etc.), global information services (GIS) data and earth mapping capability, and user interface elements. All software modules have a public framework API to support interface extensibility. This is accomplished in the extensibility and API framework layer. The module and messaging layer contains code written for single and specific purposes. This is the layer that contains user interfaces, utility classes, and messaging protocol support for communication to external software components. Finally, the application layer contains code related to executable applications such as a test-bed, utility application, or test operator console. All code is written utilizing agile software development principles, namely the SOLID principles (Single Responsibility, Open/Closed, Liskov Substitution, Interface Segregation, and Dependency Inversion) (Martin, 2012).



The Fusion visual framework is broken into six key concepts: (a) Login, (b) Layout, (c) Notification, (d) Feedback, (e) Canvas, and (f) Tiles (see Figure 3).

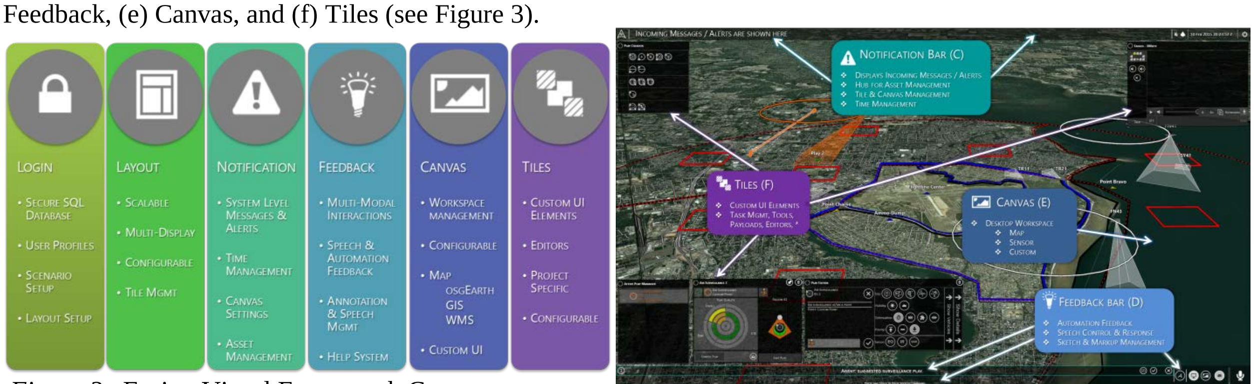


Figure 3. Fusion Visual Framework Components.

Virtual Distributed Lab

The notion of a virtual distributed laboratory (VDL) connecting various DoD and contractor sites throughout the CONUS was paramount to foster a more cohesive and distributed development and research environment. Fusion has adopted a DoD open source model, enabling joint development across a variety of projects and collaborators, all contributing to a single source repository. The core development team is located at Wright-Patterson AFB, and there are currently several offsite laboratory development teams (see Figure 4). Fusion is hosted on a secure web server and program access can be requested at https://www.vdl.afrl.af.mil/ (please contact the authors for further instructions). The software is developed on a standard Windows 8.1 PC platform in Microsoft Visual Studio and several third party developer libraries (See Figure 4). All distributed laboratory sites have similar hardware configurations and the software developers use a common set of software development and configuration management tools.

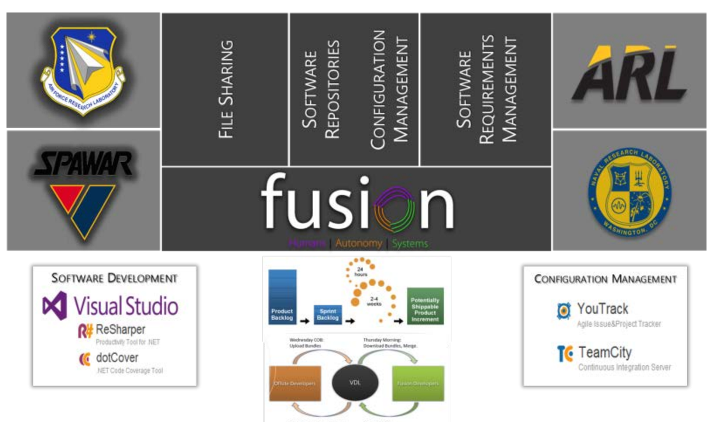
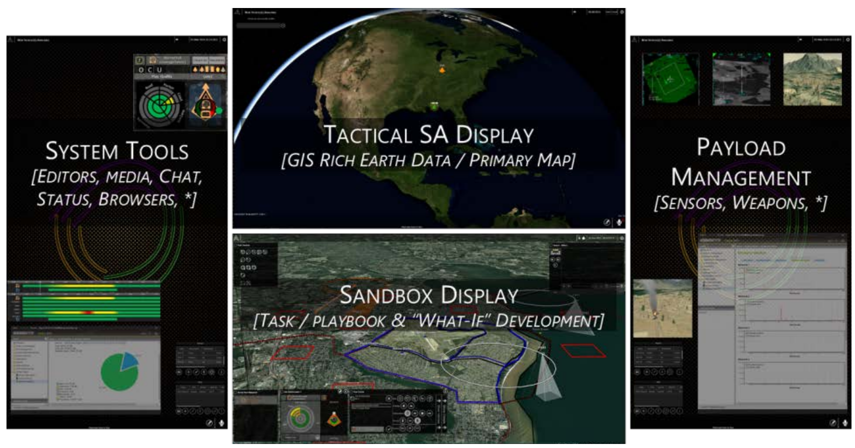


Figure 4. Fusion Virtual Laboratory Concept.

The Fusion software development team leverages SCRUM, an agile software development process. The Fusion source code repository is hosted on VDL and a strict configuration management process is followed. Once a week, offsite developers submit their changes, and the core Fusion team integrates those changes and posts a new version of Fusion on VDL for the offsite developers and researchers. In the coming year, the team will transition to a fully on-line software development cycle utilizing Git (a software configuration repository structure) and the secure Defense Research & Engineering Network (DREN). This process allows all offsite labs to keep up to date with the core Fusion team as well as keep their software well maintained. The concept of a virtual distributed laboratory has been successful due to Fusion providing a robust and flexible software architecture.

Fusion Keystone Program Application

The Fusion program formed to support two ASD R&E Autonomy Pilot Research Initiative (ARPI) projects. The purpose of this initiative was to foster research and push the envelope for autonomy-based researcg. One such project, Realizing Autonomy via Intelligent Adaptive Hybrid Control, is developing an "Intelligent Multi-UxV Planner with Adaptive Collaborative/Control Technologies" (IMPACT). This is a three year effort (FY14-FY16) with a focus on maximizing human-autonomy team agility. The first year of this effort focused on designing and implementing the user interface for higher level, goal-oriented plays (analogous to the sports ontology), which included asset management and integrating the various autonomous components. This "play" centric concept allows operators to focus on higher level goals for the vehicles, leveraging autonomous aids in accomplishing those goals, thus reducing the need for the user to direct and control vehicles manually.



The IMPACT instantiation of Fusion, as shown in Figure 5, employs a four screen layout: system tools, real time tactical situation awareness (SA) display, sandbox tactical SA display, and sensor management. The operator uses the sandbox display to perform all of their play calling and chat monitoring tasks. The other screens display tools to enhance the operator's SA.

All of the goals of the Fusion framework were critical to the success of the first year of IMPACT. Fusion, autonomous agents, external simulations, a cooperative control planner (CCA), and machine learning algorithms were combined to form a comprehensive, richly interactive environment. This was enabled by Fusion's flexible software architecture through a robust simulation environment, software extensibility, and interface instrumentation.

Flexible Software Architecture

There are four primary research threads that Fusion is addressing to accomplish the goals of developing a framework for human interaction with flexible-adaptive automation across multiple UxSs: (1) developing a software system that can generalize disparate and similar messaging protocols to be protocol-agnostic while allowing a many-to-many relationship between networked systems for the generation, distribution and consumption of network messages; (2) developing a software framework where every public element, regardless of its role as a model or user-interface element, is customizable, extendable and override-able by any other software developer in the system; (3) developing a software system that is fully instrumented to gather real-time user/machine interactions and system details for use in experimentation, software agents, and machine learning; and finally, (4) developing a software system that records the state of each of its components at a rate near 30hz and makes it user-accessible to enable discrete and continuous retrospection of the system in real-time.

Cloud-based Simulation Architecture

The development team has established an API for external software components to communicate and interact with Fusion. To date, vehicle simulations, intelligent task allocation agents, vehicle planners, speech interpreters, chat systems, sensor visualization, operator assistance components, map layer data, and monitoring components have been incorporated into the Fusion network API. These networked components employ various connection modalities (e.g., UDP, TCP/IP, ZeroMQ) and communicate using various messaging protocols (LMCP, JSON, DIS, and custom protocols). In some form, all the components are linked together in their communications modalities by use of a hub. Where appropriate, the connections and protocols are also realized into appropriate interface components in Fusion, and are intended to aid in creating a more immersive and interactive system for human-autonomy teaming.

The goal of this network API is to make the incorporation of external software as transparent and natural as possible while leveraging data efficiently. All of the instrumentation data (discussed in detail later in this section) is distributed to the communications hub, and any component that wishes to consume the data can do so with a simple subscription. Likewise, communication messages from the other components are delivered to the same hub, and Fusion (or any other component) can subscribe and receive those messages. Each of the networked components may also communicate with another networked component using this same network structure. The Fusion team has worked closely with developers of other software components in order to ensure a seamless integration. The publish/subscribe architecture present on the communications hub makes for a natural assembly: all the associated data published by any software entity is available to any other entity that needs to leverage it, thus enabling great flexibility in the potential interactions between the entities, including Fusion and its operator(s). It also establishes the framework that will be needed for our near-future extension of Fusion to a MOMU (multiple operator/multiple unmanned system) interface.

With Fusion as an operator's interface, the communication with the components are more transparent and natural. For example, in the IMPACT work that Fusion has supported, a user can define high-level goals that Fusion dispatches to an intelligent agent. The agent allocates the vehicle or vehicles to be used for the achievement of the goals and submits the realized requirement to the vehicle planner. The planner reports the plan back to Fusion, which is then shown as a candidate solution. When accepted, the plan is delivered to the vehicle simulation for execution, while another component monitors the plan execution to alert the operator in case of issues. Each of these pieces communicate using a different set of message protocols. To the operator, it appears as if all these components are part of Fusion: the operator defines a goal and approves a plan, and it appears to the operator that the deliberative activity occurs exclusively within Fusion. In this way, Fusion enhanced the operator's teaming with the autonomous system components while ensuring that the communication and feedback are transparent to the operator. Since the operator is able to define high-level goals for the IMPACT project, the system enables a realization of an enhancement in the dialog between the operator and the various software entities that provide access and control of autonomous vehicles. This enhancement was exercised in the first IMPACT evaluation, which had positive feedback from all the subjects. The flexibility and extensibility of the communications framework has provided a baseline from which the development team will further extend and enhance the human-machine teaming, creating a more immersive and flexible interactive environment.

Software Extensibility

Fusion is being used in several different projects, all of which share the goal of improving operator interactions with highly autonomous systems (with future extensions to MOMU), but have vastly different human machine interface (HMI) designs and algorithms. Due to this, Fusion was built with the goal of extensibility.

Fusion's infrastructure allows developers to override aspects of the HMI easily. Fusion adopted a layered architecture in which the framework contains the building blocks for HMI tools and services. Developers add new HMI tools and services by overriding those building blocks and developing new modules. Thus, developers can override or extend aspects of Fusion without altering the original or previous extensions. Modules can be either universal or project-specific. Through this, the user can choose which modules are loaded, and therefore affect how the Fusion user interface looks and reacts.

One example of the extensibility currently realized in Fusion is the vehicle symbol. In test beds that allow operators to control or supervise unmanned systems, vehicle symbols are important and appear in many different places in the user interface. In Fusion, vehicle symbols appear on the map, in various notifications, on the vehicle status tool, in tasking tools, in many project specific tools, and other places. Most projects have their own vehicle symbol design and it require a lot of work to replace that symbol everywhere for every project. Fusion was built with this in mind, therefore, there is a default simplistic vehicle symbol included in the framework. Every project has the ability to design and implement their own vehicle symbol. With a single line of code in the project-specific vehicle symbol code, all vehicle symbols in the entire Fusion test bed can be replaced. An additional feature is that developers do not even need to consider this during implementation: the framework handles it at run time.

Extensibility saves a great amount of development time and allows designers to try out multiple solutions. A user interface can be designed and implemented in multiple ways and, depending on which modules the user loads, a specific design is realized. This facilitates experimentation to determine the best design.

Interface Instrumentation

Data collection, agents, and machine learning all require capturing of data, which must be stored or packaged up and sent across the network. User interface interactions is one of these critical data sources. This capability was built into the Fusion framework. It is fairly non-invasive to the developers and provides a host of information, both after the fact and real time. Every user interaction, such as button clicks, typing, and mouse clicks are recorded and saved to a file.

All instrumented data is also packaged up and sent through the network to any agents, machine learning algorithms, cognitive modeling services, or other automated services that are subscribing to the data source. Instrumentation of all operator interactions is critical for human-autonomy teaming. There are many uses for this feature of Fusion. Agents use it to better understand user behavior and take or recommend actions. Machine learning employs instrumentation data to learn how individual users perform, and potentially recommend an interface change either after the fact or in real time (e.g., if a user uses certain buttons more often, the buttons can be reorganized to better suit their use). The data can inform cognitive modeling services, improving researchers' understanding of how the operators are performing.

During evaluations, all instrumentation data is recorded into a comma delimited log file. The experimenter can go back after the fact to analyze performance data. They can determine reaction times and accuracies based on the times of various user actions saved in the file. This was utilized in the IMPACT year one evaluation. For example, a chat module was employed to request tasks from the user. The chat requests and responses were instrumented, as were the user reactions. The experimenter leveraged this data to analyze how quickly and effectively the task was performed. The experimenter also noted any extra steps the operator performed, the sequence of steps taken, and the modality of their actions. All of this data is being used to analyze and improve the user interface as well as any automated services.

Human-Autonomy Dialog through Retrospection

All of the instrumentation data can be used for retrospection. Since all the data is stored, it is natural to allow it to be re-played post process or played back during runtime. Retrospection has two main applications (and potentially more): experimenters can observe what was occurring to analyze why an operator performed an action or series of actions, and operators can "pause" and "rewind" the scenario to get another look at something that occurred in the past, further enhancing the human-autonomy dialog.

The concept of an operator being able to rewind the scenario introduced the concept of a sandbox. The sandbox is an area of the user interface where the operator can invoke actions that aren't instantly carried out by the UxSs. The sandbox allows the user to evaluate autonomy-proposed actions and tweak various parameters before committing to them. Other displays within Fusion still depict current vehicle activities in real time, so the operator maintains effective SA. This can give the operator more insight into the autonomous component actions and reasoning. Another use of the sandbox is to play back the scenario using the instrumented data to see what occurred at some point in the past. This could possibly help operators make more informed, quicker decisions in the future.

Conclusion

Fusion fills a needed role in human-autonomy teaming. Since humans will remain a critical part of autonomous systems development and deployment for the foreseeable future, a clear representation and extension of an operator's intent is required. Fusion was developed to support this, and it continues to expand within the human interface role as it pertains to the Air Force Research Laboratory's goals. It is well positioned to aid in achieving many goals related to autonomy defined by AFRL, the Air Force, and the Department of Defense. Fusion directly addresses two of the four goals established by the AFRL Autonomy Science and Technology Strategy (AFRL, 2013): delivering flexible autonomy systems with highly effective human-machine teaming and creating actively coordinated teams of multiple machines to achieve mission goals. Fusion also participates in addressing two challenges from the 2010 Technology Horizons (USAF Chief Scientist, 2010) for the Air Force: highly autonomous decision-making systems and fractionated, composable, survivable, autonomous systems. Finally, the defense science board (DoD Defense Science Board, 2012) identified perception, planning, learning, human-robot interaction, natural language understanding and multi-agent coordination as key areas that would benefit from improved autonomy. Fusion has encompassed and is continuing work in five of these domains (except perception). Fusion is thus supporting and driving the emerging technologies with which the goals and challenges across all levels of the AFRL hierarchy can be satisfied.

Acknowledgements

This research also supports the "Autonomy for Air Combat Missions: Mixed human/UAV Teams (ATACM)" ARPI that is developing pilot vehicle interfaces for manned/unmanned teaming. The authors would like to acknowledge the software developed at AFRL for the Vigilant Spirit Program (Rowe, 2009) as software engineering principles and concepts have transitioned into the Fusion Framework.

References

Air Force Research Laboratory. (2013, March 5). Autonomy Science and Technology Strategy.

DoD Defense Science Board. (2012, July). Task Force Report: The Role of Autonomy in DoD Systems.

Douglass, S. A., & Mittal, S. (2013). A Framework for Modeling and Simulation of the Artificial. Andreas Tolk (Ed.), Ontology, Epistemology, and Teleology of Modeling and Simulation, Intelligent Systems Series, Springer-Verlag

Kingston, D. B., Rasmussen, S.J., & Mears, M. J. (2009). Base defense using a task assignment framework. *AIAA Guidance, Navigation, and Control Conference*, AIAA-2009-6209

Martin, Robert C., Martin, Micah (2012). Agile Principles, Patterns, and Practices in C#.

Miller, C.A., Hamell, J., Barry, T., Ruff, H., Draper, M.H., & Calhoun, G.L. (2012). Adaptable operator-automation interface for future unmanned aerial systems control: Development of a highly flexible delegation concept demonstration. AIAA Infotech @ Aerospace Conference, AIAA-2012-2529, 1-21.

Rowe, A. J., Liggett, K.K., & Davis, J.E. (2009). Vigilant Spirit Control Station: a research testbed for multi-UAS supervisory control interfaces. Proceedings of the 15th International Symposium on Aviation Psychology.

USAF Chief Scientist (AF/ST). (2010, May 15). Technology Horizons: A Vision for Air Force Science & Technology During 2010-2030, Volume 1.

Vernacsics, P. & Lange, D. (2013). Using autonomics to exercise command and control networks in degraded environments. *18th International Command and Control Research and Technology Symposium*, Alexandria, VA. DTIC ADA 587015.