

The Application of Service Oriented Software Architectures in the Fuels Treatment Community

Authors: Sim Larkin, Tami Funk, Judd Reed, Sean Raffuse, Lyle Chinkin, Mike Rauscher

Abstract

During the past decade, a proliferation of data, software systems, and analysis tools has emerged in the fire and fuels management community. Due to the heterogeneity of available data, data formats, software systems, and ad-hoc tools, awareness, access, and distribution of data and software tools in the fire and fuels community have become tremendously decentralized. Consequently, fuels treatment analysts and decision makers are left with an assortment of analysis methods, unconnected software systems (in various stages of development), and no standard approach for performing fuels treatment analysis and planning. This paper presents a discussion of how a judiciously selected set of software architecture features could be used to effectively organize the fuels treatment software models and tools to support the work of the fire and fuels management community.

Keywords

Fuels treatment, software architecture, service-oriented architecture, software systems

Introduction

During the past decade, a proliferation of data, software systems, and analysis tools has emerged in the fire and fuels management community. Due to the heterogeneity of available data, data formats, software systems, and ad hoc tools, awareness, access, and distribution of data and software tools in the fire and fuels community have become tremendously decentralized. Consequently, fuels treatment analysts and decision makers are left with an assortment of analysis methods, unconnected software systems (in various stages of development), and little guidance about the strengths and weaknesses of these systems.

The National Interagency Fuels Coordination Group (NIFCG) has acknowledged that a need exists for organization, guidance, and consistency in managing the resources available to the fuels treatment management community. Specifically, standardization and consistency in the analytical methods and approaches taught in the various fuels curricula are needed as well as an integrated, centralized software framework to manage the many data sources, models, and tools available to fuels treatment planners.

The objective of this paper is to show how other communities have addressed the issue facing the fuels treatment community and to describe how a judiciously selected set of software

architecture features could be used to effectively organize the fuels treatment software models and tools to support the work of the fire and fuels management community.

Background

The NIFCG and the interagency Joint Fire Science Program (JFSP), through both formal and informal interactions with partners and clients, became convinced that the need for an integrated software management framework is a pressing issue currently facing fire and fuels analysts and decision makers. Acting in concert with the NIFCG, the JFSP initiated the Software Tools and Systems Study (STS) in 2007 and funded Carnegie Mellon Software Engineering Institute (SEI) to perform a strategic analysis of the problem. In the spring of 2008, the JFSP funded a second phase of the STS to understand the current practices and needs of the fuels treatment planning community and to design a software architecture that supports those needs.

The STS revealed several unique characteristics of the fire and fuels management community that warrant a platform and software systems architecture that support organization and collaboration (Palmquist, 2008; Funk et al., 2008). These characteristics are described as follows:

- **A current lack of standardization and universally accepted analysis and planning methods.** The software architecture should provide a framework to organize the data and tools that fuels treatment specialists commonly use for analysis and planning but should provide flexibility in how analysts use the tools for a particular problem or situation.
- **The use of expert judgment combined with model(s) or model functions for fuels treatment planning.** It is critical that the software architecture solution supports user interaction and modularity, that is, users can independently select and utilize individual models, or functions within the system.
- **The continuous development of new models and methods.** The software architecture solution should be expandable and modular and should support the addition of new data, software models, and tools as they become available.
- **Computer administration issues.** Software installation and accessibility within many federal agencies is a concern. Therefore, the software architecture solution must be accessible without requiring the installation of proprietary software and/or other resources that may be barriers to use.

- **A current lack of interagency collaboration.** There is currently a lack of interagency collaboration although the agencies that perform fuels treatment planning face similar issues. The software architecture solution should support open analytical collaboration by allowing users to publish and share their methods and algorithms within a system library.
- **Resource limitations.** Fuels treatment planners are often responsible for many tasks beyond fuels treatment planning and do not have the time or resources required to learn, and maintain familiarity with, dozens of software models and tools. Therefore, the software architecture solution should provide analysis guidance and reporting tools to streamline fuels treatment decision-making.

A key and underlying characteristic of the fuels treatment planning community is the fact that, while there are many tools available for fuels treatment planning, no standardized and universally accepted analysis and planning approach exists. This lack of a coordinated approach is partly because of the geophysical and ecological complexities and specificities associated with fuels treatment planning and the availability of vegetation data to support analysis. As a result, fuels treatment specialists usually employ customized data and/or analytical methods combined with expert judgment for decision-making.

To address the characteristics of the fuels treatment planning community, the SEI study recommended that a platform and software architecture that supports interagency collaboration include the following key components (Palmquist, 2008):

- a software framework architecture that facilitates use and integration of data and scientific models, including a common user interface and shared data structures,
- the flexibility for users to select and compose their own data and chain of models to help address their specific analysis conditions,
- a clearly defined and articulated set of standards to allow software developers to develop models and modules that will function within the software framework architecture, and
- a lifecycle management system with processes to set priorities for software system development, training, and retirement.

The challenge of organizing and managing the many data, software models, and tools within the fuels treatment community is not unique. Many businesses and research communities have faced similar challenges organizing information, resources, and work processes to increase efficiency. As a result, a technological solution that has emerged over the past decade is the

concept of Service Oriented Architecture (SOA). In simple terms, SOA facilitates the integration of disparate software systems by separating functions into distinct units, or services, that can be made accessible across a computer network so that users can combine and reuse individual services as needed (Erl, 2005). SOA facilitates the integration of data, new software systems, and legacy software systems to streamline work processes.

An example of SOA technology is online banking where customers log in to a website hosted by their banking institution and manage their personal bank account(s) using a collection of individual services (i.e., bill pay, cash transfers, account registers). Another example is the online tax preparation service, TurboTax®, where customers can prepare and manage their personal tax information online using a set of common tools, or services. SOA is a popular and widely used architectural approach for systems development and integration to support efficiency and collaboration within a community. A key recommendation resulting from the first phase of the STS study was that the fire and fuels treatment community would greatly benefit from an SOA solution (Palmquist, 2008).

Distributed Service Oriented Architectures

A universally accepted nomenclature for characterizing software systems architectures does not exist. Furthermore, key terminology related to SOAs, including “distributed” and “collaboration”, have different meanings to different authors. For the purpose of this paper, we define “distributed computing” as computer systems working in parallel that are geographically or administratively separated. Some authors use the term “collaborator” to mean a person; others describe collaborators as computers and not the people using them. For the purpose of this paper, we use the word “collaborator” to mean a person. We further distinguish between two types of collaborators: (1) system users who collaborate with one another within a problem space to share data, processing, and analysis methods and (2) scientists who collaborate to improve the system by providing data, tools, and models to the system. We refer to these two types of collaborators as “analytical” and “structural”, respectively.

In the most general sense, distributed collaboration involves two or more geographically dispersed individuals working together to create a product by sharing and exchanging data, information, and knowledge. Collaborative environments are not software systems themselves, but they provide the framework to access and integrate data, models, and domain-specific tools to facilitate interdisciplinary collaboration.

SOA is becoming a popular and widely used approach for systems development and integration to support collaboration within a community. The benefit of effectively developed SOAs is a loose coupling of services with operating systems, programming languages, and other

technological applications ((Newcomer and Lomow, 2005). SOAs separate business or work flow functions into distinct units (services) that are made available across a network in a way that they can be combined and reused. A central control system allows the services to communicate with each other by passing data from one service to another, or by coordinating an activity between two or more services. SOA concepts are often viewed as a hybridization of distributed computing and modular programming (Erl, 2005). For the purpose of this paper, we refer to SOAs that support dynamic Internet communication as “distributed SOAs”.

From a structural perspective, distributed SOA systems generally consist of two key components:

- A **central control system** that defines and tracks service registration and facilitates service transactions through the problem domain. The control system defines the requirements (i.e., interface specifications) required for service interoperability.
- A published directory of registered **services**. Services are data and/or software module components that are platform-, programming language-, and operating system-independent.

Figure 1 provides a general diagram of the key SOA components (adapted from Panda, 2005).

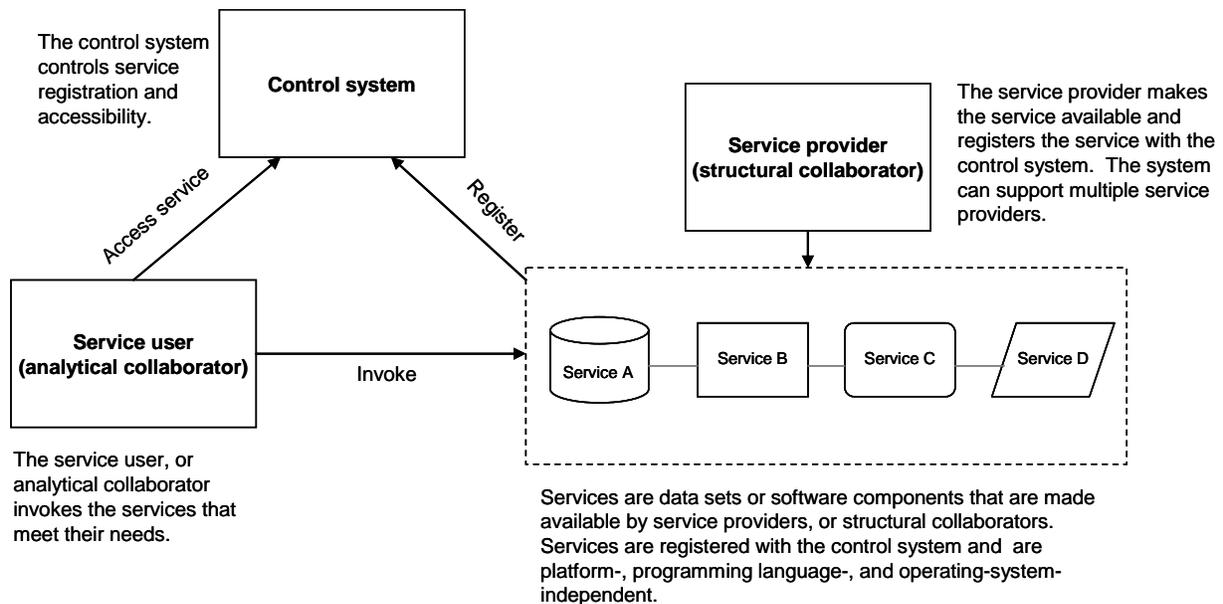


Figure1. A general diagram of the key SOA components (adapted from Panda, 2005).

An SOA acts as a framework that brings together many disparate modules (services) and functions and supports their use and reuse in a controlled manner with less burden on the user. For example, during fuels treatment analysis and planning, fuels treatment specialists often use

fire behavior models such as FlamMap¹ combined with standard Microsoft (MS) Office products such as MS Excel. Using this approach, the fuels treatment specialist must first prepare a vegetation data set, input that data into FlamMap, manually analyze the output(s) in MS Excel, and then prepare a fuels treatment prescription or burn plan. This approach can be time-consuming and requires extensive human manipulation of data, analysis expertise, and expert judgment at each step of the process. Several comprehensive standalone desktop software packages such as ArcFuels have been developed to aid fuels treatment specialists. These software systems provide tools that help integrate vegetation data with fire behavior models and provide analysis and visualization tools. While these software packages have proven to be extremely useful, they require a fair degree of knowledge of the system and software (e.g., ArcGIS), and generally offer specific fire behavior models, FlamMap in the case of ArcFuels, that are built into the system.

A well-designed and implemented SOA system would allow the fuels treatment specialist to log in to a website, have the option to use pre-loaded vegetation data or to supply their own data, choose the fire behavior model that fits their particular analysis objectives (from a suite of available models) and execute a command that would tell the control system to perform the analysis in the specified sequence, or path. The SOA system would then facilitate and automate the transfer of data and information from one step to the next to produce output information. The analyst would then view and manipulate the output data using a suite of available tools within the SOA. SOA technology can also facilitate multiple executions of the same path using a range of inputs to perform sensitivity analyses. A well-designed system would incorporate reporting tools that would help automate the development of documentation.

A key feature of SOAs, and one that is critical for the fuels treatment domain, is the ability of users to define the specific services they wish to utilize and the sequence, or chain, of operation of these services. This concept is referred to as a situational application (SA). An SA is an application that has been created for a specific situational need, designed for and developed in collaboration with a specific social network, or sub-set of system users (Watt, 2007).

A mashup is a type of SA that builds on services provided by different websites to create a new integrated experience. IBM introduced the term mashup to the software engineering world and it has since become an accepted term (Watt, 2007). A very simple example of a mashup would be a University website (independent of the Google Maps website) that utilizes Google Maps functionality to provide an interactive map of the campus (i.e., <http://fullmeasure.co.uk/mashups/ecsitemap.htm>). Another example of a mashup is zillow.com which integrates real estate tax information for a given geographic location (service A) with a map of the location (service B) to allow users to view tax information for all real estate

¹ <http://www.fire.org/index.php?option=content&task=category§ionid=2&id=9&Itemid=30>

within a particular area on a map (new integrated experience). Figure 2 expands on Figure 1 to illustrate the concept of mashups.

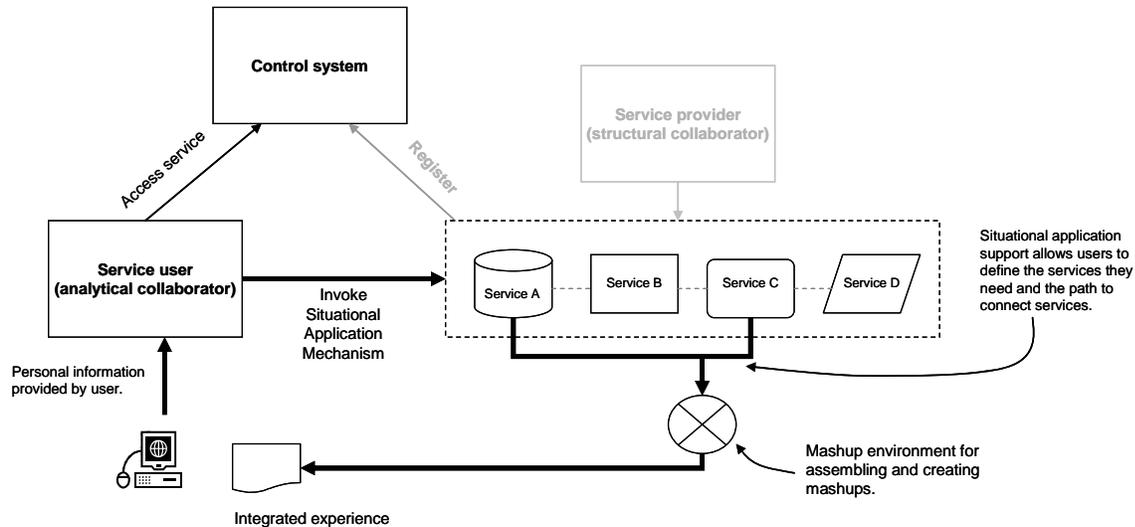


Figure 2. Illustration of the concepts of situational application and mashup technology within an SOA environment.

At least two SOA-based systems are already in use in the fire and fuels management community—the Wildland Fire Decision Support System (WFDSS) and the BlueSky Framework. The WFDSS was developed to support strategic and tactical decisions regarding real-time fire management, and the BlueSky Framework was developed to analyze and manage smoke impacts from fires.

The WFDSS combines desktop applications for fire modeling into a web-based system for easier data acquisition and provides an easy way for fire managers and analysts to accurately document their decision-making process. It organizes and manages its services to provides one standardized decision process and documentation system for all types of wildland fires. Because it is a web-based application, it facilitates analytical collaboration and sharing of analyses and reports across all levels of the federal wildland fire organization (http://wfdss.usgs.gov/wfdss/WFDSS_Home.shtml).

The BlueSky Framework combines desktop applications for fuel consumption and emissions simulations to produce estimates of emissions from fires. It consists of software framework programming code and accompanying models (services) that can be downloaded from a website and run on a local desktop machine. The Framework offers various model (service) choices at each step of a smoke impacts assessment. For example, the Framework can

facilitate the use of the Emissions Production Model (EPM)² to estimate fuel consumption and emissions. Alternatively, users may opt to use the CONSUME 3.0³ and FEPS⁴ models. In addition, the architecture supports situational applications by allowing users to start or stop processing at intermediate steps or use different pathways for different types of fires. For example, a BlueSky Framework user may opt to conclude processing after calculating emissions rather than continuing with further steps to model dispersion and ground-level smoke concentrations.

Both systems are demonstrating success in their specific user communities, and both systems currently use some of the same data sources and models used by the fuels treatment community. From a productivity perspective, these systems offer several key benefits to varying degrees:

- Standardization and organization. Both systems offer a structured and organized approach for performing analyses.
- Efficiency. Analysts no longer need to perform “overhead” tasks such as data formatting and intermediary data transfer between disparate models which can be very time-consuming.
- Increased effectiveness. Support for analytical collaboration facilitates communication and sharing of information and analytical methods between collaborators. Support for structural collaboration provides a mechanism by which new or improved software models and tools can quickly be made available to the analyst community.
- Documentation support: automated meta-data generation and document tracking facilitates the preparation of documentation to meet formal reporting requirements.
- Reduces barriers to use associated with software administration: the systems organize and manage software tools so that tools don’t have to be installed and reside on a local desktop machine.

Benefits of the SOA Approach

As noted, the fire and fuels treatment community has access to a large number of software tools and data sources (Peterson et al., 2007). What it lacks is organization and integration of these resources into a single easily accessed system. SOAs are an ideal method of addressing this need. Using an SOA approach to organize and manage the software and data resources

² <http://www1.cira.colostate.edu/smoke/epm.htm>

³ <http://www.fs.fed.us/pnw/fera/research/smoke/consume/index.shtml>

⁴ <http://www.fs.fed.us/pnw/fera/feps/>

into services and providing logical interactions among these resources (e.g. models) will more effectively facilitate fire and fuels treatment planning and decision-making. A well-designed SOA based system would automate data and processing citation, facilitate analytic collaboration, and assure uniformity of analytical methods.

Methods

SOA Systems Assessment for the Fuels Treatment Problem Area

The objectives of the JFSP's effort are to gain an understanding of how similar communities have addressed the issues facing the fuels treatment community and to describe how a judiciously selected set of software architecture features could be used to effectively organize the fuels treatment software models and tools to support the work of the fire and fuels management community. To accomplish these objectives, seven existing distributed SOA systems were identified and assessed to (1) understand how they are used within their problem domains, (2) identify the system-specific functional features that are desirable to the fire and fuels community, and (3) identify the architectural features that support those functions. The intent in assessing the example systems is to gain insights that will prove useful in the selection and design of a software architecture to support the fuels treatment community.

Summary of the Seven Selected Example Systems

The following seven systems each exhibit SOA properties. These systems were selected from both within and outside the fire and fuels domain. Each system was examined to gain insights into how it is used to support analysis and/or decision-making within its specific problem domain:

- The **Wildland Fire Decision Support System** (WFDSS) is a web-based system designed to streamline and improve decision-making processes for resource and fire management response and planning. The WFDSS is governed by the U.S. Forest Service (USFS) who also leads the responsibility for scientific development (http://wfdss.usgs.gov/wfdss/WFDSS_About.shtml).
- The **BlueSky Framework** is an open-source modeling platform that facilitates the use and interoperability of predictive models simulating the cumulative impacts of smoke on air quality from forest, agricultural, and range fires. The BlueSky Framework is governed by the BlueSky Consortium with the USFS AirFire Team leading the responsibility for scientific development (<http://www.getbluesky.org>).
- The **Integrated Forest Resource Management System** (INFORMS) is a hybrid system that is controlled by a controller interface on a local PC but runs system components

across a network. The system was designed to facilitate fuels treatment planning activities across the USFS and specifically to help support project-level National Environmental Policy Act (NEPA) analyses and landscape-level planning. INFORMS is supported by the USFS with Colorado State University (Ft. Collins) leading the responsibility for scientific development

(http://www.fs.fed.us/psw/publications/documents/psw_gtr208en/psw_gtr208en_181-184_martinez.pdf).

- The **NOAA Harmful Algal Bloom Bulletin and Mapping System (HABMapS)** is a web-based interactive mapping system with a geographic information system (GIS) that collects, stores, and displays various data layers used for detecting, monitoring, and tracking harmful algal blooms in the United States. The system was designed as a decision support tool for federal, state, and local resource and environmental managers and scientists. HABMapS is a collaborative effort supported by the National Oceanic and Atmospheric Administration (NOAA) NOAA CoastWatch, the National Ocean Service Center for Coastal Monitoring and Assessment, and the NOAA Coastal Services Center (http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20050237837_2005240380.pdf).
- The **U.S. Geological Survey Precipitation Runoff Modeling System (USGS-PRMS)** is a Unix-based, integrated modular modeling system that provides a framework to support the development, testing, evaluation, and dynamic integration of algorithms into models. The PRMS is used by river basin managers to simulate stream flow and by researchers to model hypothetical stream flow scenarios. The system allows users, or analytical collaborators, to create situational applications for specific areas and watersheds. It allows researchers, or structural collaborators, to develop and publish new models and subroutines. The PRMS is supported by Center for Advanced Decision Support for Water and Environmental Systems (CADSWES) at the University of Colorado and the USGS (<http://water.usgs.gov/software/PRMS/>).
- The **Federated Data System (DataFed)** is a web-services based software framework architecture that facilitates collaboration among autonomous, distributed data providers and users in the air quality analysis community. DataFed facilitates registration of distributed data (on remote servers) into a centralized catalog and provides a basic set of tools for data exploration and analysis (aggregation, summary, visualization, etc.). All users and data providers have access to all data sources and access is open to the public. The DataFed infrastructure was developed by the Center for Air Pollution Impacts and Trends Analysis (CAPTIA) at Washington University in St. Louis with funding from the National Science Foundation and NASA (<http://datafed.net>).

- The **Global Earth Observation System of Systems (GEOSS)** is a web-based software framework consisting of a portal, clearinghouse, and registry system for the exchange and dissemination of earth observation data. The purpose of GEOSS is to achieve comprehensive, coordinated, and sustained observations of the Earth system, to improve monitoring of the state of the Earth, increase understanding of Earth processes, and enhance prediction of the behavior of the Earth system. The GEOSS is a fully collaborative and open system in that all users and data providers have access to all data sources and access is open to the public. The GEOSS infrastructure is supported by the U.S. Environmental Protection Agency (EPA) (<http://www.epa.gov/geoss/>).

All seven systems, to varying degrees, are distributed SOAs and provide data and/or tools for analytical collaboration.

Key Functional Features for the Fuels Treatment Problem Area

Each of the seven systems serves a unique purpose and embodies a unique set of features. However, most of them have a number of important features in common, and many of these features are significant to the fire and fuels community. Each system contains services that include data transformations, software models, and analysis functions.

The first step in developing a list of functional features by which to compare the example systems was to identify the system quality attributes that are most important for the fuels treatment domain. Quality attributes are characteristics of a system that are required for success and they describe the design implications for a set of attributes in the context of SOA's. SEI has developed a somewhat standard list of quality attributes that are used as part of formal software architecture review and design (O'Brien et al., 2005). Through conversations with the JFSP and members of the fire and fuels community, a subset of the quality attributes developed by SEI was identified as key to the fuels treatment community:

- **Modularity** – each service (i.e., software models, data sets, and tools) is modularized or broken down into atomic processes to allow users of the system to run one process or service independent of other services.
- **Extensibility** – the system can be expanded over time to support the incorporation of new tools and data as they become available.
- **Flexibility** – the system is flexible to allow users to customize data and model execution to fit their specific situational analysis. The system supports situational applications.
- **Portability** – the system is easy to access and use from any standard desktop computer and does not require proprietary software or systems.

- **Ease of use** – the system is straightforward and practical to use through a well-designed interface. Specialized training or software skills are not required to run the system.

Based on the quality attributes above, a list of functional characteristics that support each attribute was developed (Table 1). The seven example systems were examined to determine the presence or absence of each functional characteristic. It is important to note that each system may exhibit different functional characteristics to varying degrees; however, in this assessment, if a system exhibits a characteristic to any degree, it was noted. Table 2 lists each system and indicates the presence (indicated with an “x”) or absence (indicated as blank) of each functional characteristic.

Table 1. List of functional characteristics that support SOA key quality attributes for the fuels treatment domain.

System Quality Attribute	Functional Characteristic that Supports Quality Attribute
Modularity	a) System components (services) can be run independently. b) The system can be stopped or started at any step in the overall process.
Extensibility	c) Structural collaborators can add services to the system via a governing body. d) Structural collaborators can add services to the system dynamically.
Flexibility	g) System contains pre-defined or default pathways through the system. h) Analytical collaborators can create situational applications or mashups. i) System has mechanism to perform sensitivity analyses. j) The system is smart enough to recognize user error and explain alternate actions. k) The system has built-in scientific error-checking capabilities; that is it identifies and filters out scientifically unreasonable results.
Portability	l) The system can be run without the use of specialized or proprietary software from a standard desktop computer.

Ease of Use	m) The system has a straightforward user interface and does not require knowledge of specific software packages or programming language(s).
-------------	---

		BlueSky Framework	Wildland Fire Decision Support System (WFDS)	INFORMS	NOAA - Harmful Alga Bloom Forecasting Decision Support Tool (HabMap)	USGS Precipitation Runoff Modeling System (PRMS)	CAPITA DataFed	EPA Global Earth Observation System of Systems (GEOSS)
Modularity	System components (services) can be run independently.	X	X	X	X	X	X	X
	The system can be stopped or started at any step in the overall process.	X	X	X	X	X	X	X
Extensibility	Structural collaborators can add services to the system via a central governing body or organization.	X			X	X	X	X
	Structural collaborators can add services to the system directly through a service registration system.	X			X	X	X	X
Flexibility	System contains pre-defined or default pathways through the system.	X		X	X	X		
	Analytical collaborators can create situational applications or mashups.	X		X	X	X	X	
	System has mechanism to perform sensitivity analyses.		X	X	X	X		
	The system can recognize user error and explain alternate actions.	X	X	X	X			
	The system has built-in scientific error-checking capabilities; that is it identifies and filters out scientifically unreasonable results.				X	X		
Portability	The system framework does not rely on the use of specialized or proprietary software.	X				X	X	X
	The system can be run without the use of specialized or proprietary software from a standard desktop computer.	X	X	X		X	X	X
Ease of Use	The system has a straightforward user interface and does not require knowledge of specific software packages or programming language(s).	X	X	X	X	X	X	X

Note that ESRI ArcGIS software is considered standard software because most fire and fuels specialists have access to and use it.

Table 2. Functional characteristics exhibited by each example system. The presence of a feature is indicated with an “x” and the absence is indicated as blank.

During the systems assessment, it was discovered that while the CAPITA DataFed and GEOSS systems are SOA-based, they are mainly used for data access, visualization, and exploration and are somewhat inherently different in that the services are more data-oriented and less software model-oriented. Because the software architecture to support the fuels treatment domain must have the ability to organize and manage software models, we will focus the remainder of our discussion on the other five systems.

As shown in Table 2, all the systems selected are modular. Modularity would be expected considering that it is a cornerstone feature of any SOA. The systems do vary in how they support modularity, that is, some are distributed systems and some are not. All the systems, except WFDSS and INFORMS, are extensible in that they support structural collaboration. Structural collaboration is an important feature for the fuels treatment problem space because new data, software models, and tools (services) are always under development. Therefore, the system should support structural collaboration. Furthermore, dynamic structural collaboration is desirable because it would allow model developers to register their services directly to the system and the newly added services would dynamically become available to the analytical collaborators. This approach also reduces the overhead involved in maintaining the system and keeping it current as new tools are developed.

The extent of flexibility of each system and the degree of SA support varies widely because the importance of SAs is domain-specific. The BlueSky Framework, INFORMS, the NOAA HabMap system, and the USGS PRMS, which are all used for area-specific, mid- to long-range analysis and planning, make significant use of SAs. In contrast, the WFDSS architecture minimally supports them. The WFDSS problem domain requires real-time, fire-specific analyses; therefore, the ability to create an SA for a specific fire situation and to reuse it is of little use. In addition, because every fire situation is different, WFDSS analysts make use of software tools but rely heavily on expert judgment to determine the most appropriate response to individual fire events.

Least common to the systems we assessed are built-in functions that perform error checking, that is, routines or filters that are built into the system to recognize user error and provide alternate actions. Error-checking functions ensure that the user is utilizing the services within the system in a way that makes sense. Error-checking functions are important when analyses are being performed to support regulatory requirements or processes. In the case of fuels

treatment planning, error-checking functions would assist analysts in conforming to NEPA requirements when developing fuels treatment plans.

All systems, with the exception of the NOAA HabMap system, are portable in that they can be run without the use of specialized or proprietary software from a standard desktop computer. The NOAA HabMap system was developed with proprietary architectural and visualization software developed by Applied Coherent Technologies (ACT) and requires use licenses. Note that for this assessment the ESRI suite of GIS software was not considered proprietary as most fuels treatment analysts have access to and experience with ESRI products.

Ease of use is a somewhat subjective characteristic but is loosely defined to mean that a system is straightforward and practical to use through a well-designed interface, and specialized training or software skills are not required to run the system. While all the systems require some level of domain-specific knowledge and skill, our assessment centered on the question, could a typical user with some expertise in the problem domain understand and use the system in a few hours? All systems appear to be fairly straightforward to use; however, their usability may vary substantially. Usability is a measure of the quality of a user's experience interacting with information or services within a system. We did not comment on the usability of the systems because this characteristic is heavily influenced by system implementation rather than architectural design.

One challenge we encountered in examining the systems is that they are undergoing rapid development. Features that we found lacking in the versions we examined may be added in subsequent releases. In interviews with the systems' proponents, we had to be very careful to distinguish between abilities that are included in the architecture and those that have actually been implemented into accessible systems.

Architectural Approaches that Support the Functional Features Needed in the Fuels Treatment Problem Area

To gain deeper insight, we examined the software architectures of each system to understand how the specific functional features of interest have been realized. One of the key questions that we sought to explore is how applicable or portable the example systems' architectures are to the fuels treatment problem space. A very flexible and portable system would, naturally, be more desirable than one in which features unique to one problem space are inextricably built into the architecture. Likewise, systems that have been built to solve problems similar to those of fuels treatment planning are of particular interest.

The NOAA HabMap system is based on a proprietary framework. This framework is very competent, powerful, and flexible and it supports SAs through a top-down scripting language or through a results-oriented (bottom-up) data layer chaining mechanism. Services may be

selected from a tool kit of basic image processing functions or from custom applications. The services may be located on the main system or distributed across a network. The system's manufacturer cited numerous problem domains where this framework had been used. Yet, in spite of all its positive attributes, this system suffers from a few major defects which may be too insurmountable for the framework to be applied directly to the fuels treatment problem domain. One obvious defect is the architecture's reliance on a proprietary framework which would most likely be prohibitively difficult to deploy across the fuels treatment community.

WFDSS is similarly based on a proprietary and custom-developed framework. But, in contrast to the NOAA HabMap system, it is hosted exclusively by a centralized node. This node has a remarkably powerful array of computers dedicated to providing WFDSS services. While it lacks significant features necessary for hosting SAs, it does seem to have an effective framework for hosting a large number of diverse services and managing the flow of data among these services.

In contrast to the large, centralized server approach used by WFDSS, the BlueSky Framework is an SOA-based control system (referred to as a kernel) and a set of services (models and other programs) that are delivered as an integrated package. It can be downloaded and installed on a collaborator's computer free of charge. The system has recently been extended into a network-enabled system that could be hosted by a large server and accessed by a community of users. The BlueSky Framework supports analytic and structural collaboration. To add a service to the Framework, structural collaborators must create a "wrapper" for their models. The wrapper contains information associated with the software model that allows it to communicate with the Framework. The BlueSky Framework is built on the programming language Python which was selected by the Framework developers specifically for its ease of use and portability.

The USGS PRMS system is based on an architectural framework called the Modular Modeling System (MMS) (Leavesley et al., 1996). The MMS is an integrated system of Unix-based software that users can download from a website and run locally on a standard desktop computer. This system includes an interactive model builder interface called Xmbuild that allows analytical collaborators to create a chain of data and models that can then be executed to complete an analysis. This framework has well developed visualization tools and sensitivity analysis tools.

Findings and Discussion

Following are summaries of the key findings resulting from this assessment:

- To some degree, five of the seven systems assessed support analytical and structural collaboration.

- While all the systems have the ability to incorporate new data and models as they become available, the degree of automatic, dynamic, and open structural collaboration is varied.
- All the systems are SOA-based; however, the architectural implementation of each system is very different.
- Among the systems assessed, inter-service communication methods in use vary widely. Consequently, it is not possible to interchange services among the current systems without some degree of development and customization which ultimately impedes structural collaboration by confining collaborators to small isolated domains. The community at large would benefit from the acceptance of uniform service definitions and inter-service communication formats

While several architectural approaches were identified and explored for application in the fuels treatment domain, there is a general lack of adherence to SOA systems standardization. Because of the lack of standardization, services and data cannot be interchanged among systems without substantial effort, that is, one existing architecture cannot simply be applied to the fuels treatment domain without significant development effort to adapt it to meet the needs of the domain.

The lack of well-developed and standardized data history tracking and additional metadata storage is surprising. There are U.S. and international (ISO19115) standards for storing and reporting metadata. The U.S. Federal Geographic Data Committee (FGDC) defined the Content Standard for Digital Geospatial Metadata in 1994. This standard was updated in 1998. According to Executive Order 12096, all federal agencies must use this standard to document geospatial data created as of January 1995. All U.S. government units are required to adhere to the CSDGM when documenting and distributing spatial data. Many state and local governments have adopted this standard as well. Yet, few of the systems examined seem to conform to this standard. A dedicated effort to adopt both the required and optional fields in the FGDC metadata standard would constitute significant progress toward the fire and fuels communities data history and authorship-tracking requirements.

Encouraging developments indicate progress in this direction. The Open Geospatial Consortium (OGC) is an international organization of private industry and government that has proposed standards for encoding geospatial data. Adoption of OGC standards, as well as participation in the organization itself, is voluntary. Yet, there is steady progress toward the adoption of OGC formats for encoding data passed among the services of SOA-based systems used by the fire community. Use of OGC formats alone does not enable the systems to interoperate. But it does make it easier for them to do so. At a minimum, the use of standardized data formats makes it easier to use the output from one system as the input to another.

This examination of multiple systems with varying SOA designs exposed a need for new technologies that will facilitate collaboration among the fuels treatment planning community as well as collaboration between the users of the fuels treatment decision support system and each of the similar systems (i.e., WFDSS and the BlueSky Framework) that are currently in use. It is predictable that the trend toward increased use of distributed computing techniques will facilitate fire and fuels treatment community users' inter- and intra-system collaboration. Continued emphasis on modularity, portability, and flexibility will be critical to success in this area. Ideally, the entire fire and fuels community would benefit from the development of three systems—WFDSS, the BlueSky Framework, and the fuels treatment system—that could be intra-operable and would leverage the services of each other to eliminate service redundancy among systems. To facilitate system intra-operability, a set of standardized data formats, service-interface specifications, and input and output requirements should be established. If each system adhered to the same set of standards and requirements, intra-operability could be achieved.

A system that meets the fire and fuels treatment community's needs must have a very well developed SA editor and a robust suite of tools to support the iterative development and improvement of SAs. It will have to support relatively complex SA structures. Many of the assessed systems have little or no support for complex SAs. Others have some SA support but in a manner that does not support iterative or sensitivity analysis. The fuels treatment system will have to have a relatively sophisticated SA development tool that includes support for both

supervised and unsupervised interactive processing loops. It must be possible to monitor scripts as they are in progress and to review intermediate results in a geographic data browser or a GIS.

The services must include or be accompanied by data quality assurance components and authorship tracking. It is important for users to share data with other users of the system or with users of other systems. Such collaboration must be accompanied with authorship tracking so that users can be confident that data are properly attributed. Another feature of the fire and fuels treatment community application space that is less pronounced in the domains of the other systems is the need for careful document and method tracking. For example, there is a need to record, within each data product's "meta-data" a history of how, and by whom, that product was created.

Due to its long-range strategic application, this decision support system does not require the same responsiveness of other real-time systems. Support for overnight (or longer) execution of large, complex simulations may be required. This kind of support has ramifications for system design that include the need to address session interruptions in "off line" or "batch mode" SA execution. Ideally, the system could analyze a suite of proposed strategies and compare the probable outcomes of each.

Finally, the new system must be accessible from standard computers. It has not yet been determined where the services will actually reside and where they will run. This ambiguity of specification is not problematic if the system and its individual components are designed with an appropriate degree of modularity and portability.

While all the systems assessed possess useful features, no one architectural approach can be easily applied to the fire and fuels domain. Several key features of each system have been identified, and the intent is to emulate these features in the design and development of a fuels treatment decision support system. These architectural features include

- Open and dynamic structural collaboration mechanisms. Through carefully developed and well-documented standardization of service interfaces, structural collaborators are

able to develop new services that can be easily added to the architecture through the use of a service registry and support for distributed web-services.

- A powerful server hosting array. A strong operational computing system would serve the fuels treatment community well. Powerful computing systems and substantial data storage capacity allow users to perform computationally complex analyses remotely from their desktop computers, even in the absence of high-bandwidth network connectivity.
- A well-developed model management system. A model management system that supports the development of complex situational applications is necessary to support the fuels treatment domain and the need for SA authoring, reuse, and analytical collaboration.

Conclusions and Recommendations

The fire and fuels treatment community would benefit from an SOA-based decision support system that organizes and manages the plethora of disparate data and software systems now in use. A well-designed SOA-based system would automate data and processing citation, facilitate analytic collaboration, and assure uniformity of analytical methods. Overall efficiencies in the fire and fuels management domain could also be achieved if the fuels treatment decision support system were designed to communicate and share services with other SOA-based systems currently in use in the broader fire community.

The system should be modular, extensible, flexible, portable, and easy to use. To achieve these goals, the system should include a framework that provides standardization and guidance for fuels treatment analysis and planning and supports situational applications. Ideally, the framework would integrate common elements (services) of related systems in the fire and fuels domain, namely, WFDSS and the BlueSky Framework. The system must run on, or at least be accessible from, standard desktop computers and should not require the use of specialized or proprietary software. It should include data visualization and manipulation tools that are easily accessible and should leverage the capabilities of existing commonly used programs (i.e., web-browsers, Google Earth, ArcGIS, Geographic Resource Analysis & Science [GRAS], etc.).

Of the five key quality attributes important for the fuels treatment domain, the most straightforward to implement are modularity, portability, and ease of use. All SOAs are inherently modular. Highly portable systems are generally architected using open source programming language(s) and/or provide system access via a standard web-browser so that users are not burdened with the need for proprietary software. Ease of use can be achieved by designing and developing an intuitive and effective user interface tailored specifically to the needs of the problem domain.

From an implementation perspective, extensibility and flexibility are the more challenging to achieve. Extensibility facilitates structural collaboration and the degree to which service providers can openly and dynamically make their services available to the analytical community. The choice of architectural approach for the fuels treatment domain should accommodate open and dynamic structural collaboration through the use of a well-defined and articulated set of collaboration standards (i.e., data formats, interface specifications, meta-data). Furthermore, these standards should be adopted throughout the fire and fuels domain so that other decision support systems such as WFDSS and the BlueSky Framework can share common services and leverage the benefits of one another. Success will require collaboration among the system development teams and adoption of a common set of standards.

System flexibility is also a challenging attribute to achieve relative to the other quality attributes and particularly, the support for situational applications. The implementation of situational applications requires a sophisticated mechanism that allows analytical collaborators to interact with the system in a way that they specify. Mashup technology should be utilized to support the type of situational applications required by the fuels treatment problem domain.

In summary, the fire and fuels community is making positive progress in the direction of organization and management of the various tools available to the community by utilizing SOA technology. The development of an SOA-based fuels treatment decision support system would begin to create consistency among the systems already in use. The broader community would benefit from the development and implementation of a common set of standards that allow

fire and fuels systems to communicate with one another and leverage common services moving forward.

References

- Erl T. (2005) *Service-oriented architecture: concepts, technology, and design*, Prentice Hall PTR, Upper Saddle River, NJ.
- Funk T.H., Rauscher M., Raffuse S.M., and Chinkin L.R. (2008) Findings of the current practices and needs assessment for the Interagency Fuels Treatment Decision Support System (IFT-DSS) project. Technical memorandum prepared for the The Interagency Fuels Treatment Work Group (IFTWG), by Sonoma Technology, Inc., Petaluma, CA, STI-908038.01-3504, November.
- Leavesley G.H., Markstrom S.L., Brewer M.S., and Viger R.J. (1996) The modular modeling system (MMS) — the physical process modeling component of a database-centered decision support system for water and power management. *Water, Air, & Soil Pollution* **90** (Numbers 1-2), 303-311. Available on the Internet at <<http://www.springerlink.com/content/v054020t506w2882/fulltext.pdf>>.
- Newcomer E. and Lomow G. (2005) *Understanding SOA with web services*, Addison-Wesley Professional.
- O'Brien L., Bass L., and Merson P. (2005) Quality attributes and service-oriented architectures. Technical note prepared by Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, CMU/SEI-2005-TN-014, September. Available on the Internet at <<http://www.sei.cmu.edu/pub/documents/05.reports/pdf/05tn014.pdf>>.
- Palmquist M.S. (2008) Working summary of the SEI's engagement with the Joint Fire Science Program. Report prepared for the U.S. Department of Defense by the Acquisition Support Program, Software Engineering Institute, Carnegie Mellon University, April.
- Panda D. (2005) An introduction to service-oriented architecture from a Java developer perspective. Available on the Internet at <<http://www.onjava.com/pub/a/onjava/2005/01/26/soa-intro.html?page=1>>. January 26.
- Peterson D.L., Evers L., Gravenmier R.A., and Eberhardt E. (2007) Analytical and decision support for managing vegetation and fuels: A consumer guide. General technical report prepared by U.S. Department of Agriculture, Forest Service, Pacific Northwest Research Station, Corvallis, OR, PNW-GTR-690, January. Available on the Internet at <<http://www.wy.blm.gov/fireuse/pubs/AnalyticalDecisionSupport.pdf>>.
- Watt S. (2007) Mashups -- The evolution of the SOA, Part 2: situational applications and the mashup ecosystem. Available on the Internet at <<http://www.ibm.com/developerworks/webservices/library/ws-soa-mashups2/>>. November 8.