INTERAGENCY FUELS TREATMENT DECISION SUPPORT SYSTEM SOFTWARE DESIGN SPECIFICATIONS

Software Specifications
STI-909029.03-3664-SS

By:
Neil J.M. Wheeler
Judd E. Reed
Kevin D. Unger
Sean M. Raffuse
Steven A. Ludewig
Tami H. Funk
Eric A. Gray

Sonoma Technology, Inc.
1455 N. McDowell Blvd., Suite D
Petaluma, CA 94954-6503

Prepared for:
The Joint Fire Science Program
3833 South Development Avenue
Boise, ID 83705

August 18, 2009

ACKNOWLEDGMENTS

Many individuals have contributed to this software design specifications document for the Interagency Fuels Treatment Decision Support System (IFT-DSS). We are greatly appreciative of the support and guidance provided by John Cissel, the Joint Fire Science Program (JFSP) program manager. Erik Christiansen, Chair of the National Interagency Fuels Coordinating Group (NIFCG), provided key help and support for analyzing the fuels treatment process that specialists from all agencies struggle with. The JFSP Fuels Treatment Working Group (FTWG)—Michael Beasely, Dennis Dupuis, Mark Finney, Glen Gibson, Randi Jandt, David Peterson, Tessa Nicolet, and Brad Reed—has guided our work and represented the first line of critique and innovative ideas for the project. The JFSP Software Tools and Systems Study Advisory Committee—Pat Andrews, Nate Benson, Mike Hilbruner, Mike Hutt, David Peterson, Carol Saras, Paul Schlobohm, Shari Shetler, and Tim Swedberg—has kept the study headed in the right direction and made sure that all of the various components of a successful solution were considered.

We would like to specifically acknowledge the fuels treatment specialists who have agreed to participate in this effort to serve as the IFT-DSS proof of concept (POC) Test User Group—Randi Jandt, Brad Reed, Tessa Nicolet, Sean McEldery, Jon Wallace, Mack McFarland, Brenda Wilmore, Jim Roessler, Gwen Lipp, Perry Grissom, Eric Miller, Karen Folger, Nikia Hernandez, Gary Curcio—who have provided, and will continue to provide, valuable feedback regarding the functionality and usability of the IFT-DSS. We also thank the broader group of 49 field fuels treatment specialists who helped develop and refine the fuels treatment decision support process.

We would like to acknowledge our appreciation of the fire and fuels science and software development community—the Fire and Environmental Research Applications (FERA) team of Eric Twombly, Mark Finney, Joe Scott, Alan Ager, Nick Crookston—for their cooperation and feedback related to integrating data and software applications that will be a part of the IFT-DSS. We would also like to acknowledge the managers of the Wildland Fire Decision Support System (WFDSS)—Tom Zimmerman, Rob Seli, and their development team—and the BlueSky Framework—Sim Larkin—for their willingness to work collaboratively to develop software systems that can communicate with one another to create efficiencies in the fire and fuels domain.

We would also like to thank the Information Technology (IT) specialists who helped ensure that we have considered agency IT requirements—John Gebhardt, John Noneman, Laura Hill, and Joe Frost. All of these groups constitute the large group of stakeholders whose jobs will be positively affected by the results of this project. It will take long-term attention and energy from these groups of stakeholders and others, yet to be engaged, to create an effective community of interest that will ensure that the IFT-DSS helps all of us do our jobs better.

Finally, we would like to thank the other members of Sonoma Technology's (STI) IFT-DSS design and development team who contributed to the IFT-DSS design, including Liron Yahdav, Alan Healy, Jason Amador, and Stacy Drury.

VERSION CONTROL

Draft Version 1	July 29, 2009
Draft Version 2	August 18, 2009

TABLE OF CONTENTS

Section **Page**

ACKNOWLEDGMENTS iii
LIST OF FIGURES vii
LIST OF TABLES ix
GLOSSARY xi
EXECUTIVE SUMMARY ES-1
ES.1 Introduction ES-1
ES.2 The Fuels Treatment Planning Process ES-1
ES.3 IFT-DSS Software Architecture ES-2
ES.4 IFT-DSS POC Functionality and Scientific Models ES-4
ES.5 IFT-DSS User Experience ES-5
ES.6 IFT-DSS Software Design ES-6

1. INTRODUCTION 1-1
1.1 Purpose 1-2
1.2 Document Organization 1-3
1.3 IFT-DSS Naming Convention 1-3
1.4 References to Other Relevant Documents 1-4

2. SYSTEM OVERVIEW 2-1
2.1 Service Oriented Architecture 2-1
2.2 Rationale for a Service Oriented Architecture Approach 2-2
2.3 IFT-DSS Users and Stakeholders 2-3
2.4 Overview of the Fuels Treatment Decision Support Process 2-4
2.5 IFT-DSS POC Requirements 2-6
2.6 Overview of the IFT-DSS Architecture 2-7
2.6.1 Architectural Components 2-7
2.6.2 Key Architectural Features and Functions 2-8
2.7 IFT-DSS POC Functionality 2-9
2.7.1 Data and Model Implementation 2-9
2.7.2 IFT-DSS POC Work Flow Scenarios 2-10
2.7.3 IFT-DSS POC Implementation 2-15
2.8 IFT-DSS POC Model Integration 2-16
2.9 IFT-DSS Connections to Other Systems 2-17

3. THE IFT-DSS POC USER EXPERIENCE 3-1
3.1 Logging into the IFT-DSS 3-1
3.2 Phase I – Project Setup and Planning 3-3
3.3 Phase II – Software Model Execution and Iterative Analysis 3-5
3.4 Phase III – Project Finalization, Documentation, and Archive 3-9

Section **Page**

4. TECHNICAL SOFTWARE DESIGN 4-1
   4.1 Components 4-3
      4.1.1 Models 4-4
      4.1.2 Model Adaptors 4-6
      4.1.3 Geodatabase 4-13
      4.1.4 Data Sequencer 4-14
      4.1.5 Executive 4-16
      4.1.6 Scenario Database 4-19
      4.1.7 Navigator 4-23
      4.1.8 Project and Planning Database 4-24
      4.1.9 Planner Web Application 4-25
   4.2 Interfaces 4-26
      4.2.1 Model Data Exchange Interface 4-27
      4.2.2 Model Control Interface 4-28
      4.2.3 Sequencer Control and Monitoring Interface 4-29
      4.2.4 Executive Scenario Polling and Feedback Interface 4-31
      4.2.5 Executive Launching and Progress Reporting Interface 4-31
      4.2.6 Map and Data Presentation Interface 4-32
      4.2.7 Web Page Generator and Database Interface 4-33
   4.3 Behaviors and Interactions 4-33
      4.3.1 Project Workflow 4-34
      4.3.2 Executive Scenario Compilation 4-34
      4.3.3 Executive Orchestration of Model and Geodatabase Interaction 4-35
      4.3.4 User Interaction with the Graphical User Interface and Database 4-36
      4.3.5 User Interaction with the Web Map Service and the Geodatabase 4-38

5. TECHNOLOGIES 5-1
   5.1 Models and Model Adaptors 5-1
   5.2 Geodatabase 5-2
   5.3 Data Sequencer and Executive 5-2
   5.4 Planning and Scenario Databases 5-3
   5.5 Navigator and Planner 5-3

6. REFERENCES 6-1

LIST OF FIGURES

Figure **Page**

ES-1 Illustration of the five key architecture components of the IFT-DSS.....ES-3

ES-2 Data, transformations, and scientific models to be implemented in the IFT-DSS POC system.....ES-4

ES-3 Mock-up of the IFT-DSS POC project setup screen. ES-5

ES-4 Overall structure of the IFT-DSS depicting the system's major components and software interfaces ES-7

1-1. Naming convention for the IFT-DSS..... 1-4

2-1. Work flow diagram illustrating how the fuels treatment planning work flow scenarios fit together from a process perspective. 2-5

2-2. Illustration of the five key architecture components of the IFT-DSS. 2-8

2-3. Data, transformations, and models to be implemented in the IFT-DSS POC system..... 2-10

2-4. Process diagram for the data acquisition and preparation work flow scenario to be implemented in the IFT-DSS POC. 2-11

2-5. Process diagram for the strategic planning workflow scenario as it will be implemented in the IFT-DSS POC. 2-12

2-6. Process diagram for the prescribed burn planning work flow scenario using the FlamMap pathway in the IFT-DSS POC. 2-13

2-7. Process diagram for the prescribed burn planning work flow scenario using the FCCS pathway in the IFT-DSS POC..... 2-14

2-8. Illustration of the pathways and implementation order for the IFT-DSS POC. 2-15

3-1. Mock-up of the IFT-DSS user login screen..... 3-1

3-2. Screen shot of the IFT-DSS POC user home page. 3-2

3-3. Screen shot of the "My Profile" screen where users manage their profile information..... 3-2

3-4. Mock-up of the user options available on a user's home page..... 3-3

3-5. Mock-up of the IFT-DSS POC project setup screen. 3-4

3-6. Mock-up of an example modeling scenario graph in the IFT-DSS POC. 3-6

Figure **Page**

3-7. Example results of fireline intensity from FlamMap..... 3-7

3-8. Example page showing input options to model fire behavior for a range of weather conditions using FlamMap..... 3-8

3-9. Example page for exploring fire effects modeling results for a range of weather conditions using Consume 3-9

3-10. Example Manage Projects screen in the IFT-DSS POC with two project analyses displayed. 3-10

4-1. Overall structure of the IFT-DSS depicting the system's major components and software interfaces. 4-2

4-2. Illustration of the three model hosting methods by which models can be integrated into the IFT-DSS and function as services. 4-7

4-3. UML diagram of the model communication infrastructure..... 4-9

4-4. Conceptual ERD for the Scenario Database..... 4-20

4-5. Conceptual ERD for subset of the Scenario Database to facilitate efficient model execution. 4-21

4-6. Conceptual ERD of user interface HTML templates 4-22

4-7. Subcomponents of the Navigator. 4-23

4-8. Conceptual ERD for Project and Planning Database. 4-25

4-9. Overview of run execution. 4-34

4-10. Swim lane diagram of the Navigator, Executive, and Scenario DB interaction. 4-35

4-11. Swim lane diagram of Executive, Data Sequencer and GDB, and models interaction... 4-36

4-12. Swim lane diagram of the user, Navigator (GUI), page creator, and scenario database interaction..... 4-37

4-13. Swim lane diagram of the user, WMS, and GDB interaction. 4-38

LIST OF TABLES

Table **Page**

1-1. IFT-DSS proposed naming convention during development. 1-4

2-1. IFT-DSS requirements and how they will be realized in the POC. 2-6

4-1. Representation of an array of grid cells containing elevation values. 4-5

4-2. Pseudocode for an example slope/aspect model. 4-5

4-3. Pseudocode for key member functions to the output stream class. 4-10

4-4. Pseudocode of the readObject member function of the input stream class. 4-11

4-5. Pseudocode of the model hosting program. 4-12

4-6. Brief pseudocode for the Executive program. 4-16

4-7. Pseudocode for compiler component of the Executive. 4-18

4-8. Messages between the Executive and Model Adaptors. 4-29

4-9. Messages between the Executive and Data Sequencer. 4-30

GLOSSARY

TERM	DESCRIPTION
API	Application programming interface
Area of interest (AOI)	For IFT-DSS, a scale-independent unit of area defined by a user. Within an area of interest, project areas and vegetation units can be defined for analysis. There is no minimum size unit but a maximum unit will be limited to one million acres (approximately 400,000 hectares)
BlueSky Framework	An SOA system to facilitate predictions of smoke emissions and air quality impacts from fires
Consume	A fire effects prediction model. It uses fuel loadings, fuel moisture, and weather variables to predict fuel consumption, particulate emissions, and heat energy released under prescribed fire and wildfire conditions.
DAG	Directed Acyclic Graph (used to represent scenarios in the IFT-DSS)
.dll	dynamic link library
ERD	Entity Relationship Diagram
Executive	The component that compiles scenario segments and directs the execution of models
FCCS	Fuel Characteristic Classification System
FOFEM	First Order Fire Effects Model; a set of fire effects prediction models. FOFEM uses fuels and vegetation information to provide estimates of fuel consumption, tree mortality, soil heating, and particulate emissions.
FRAMES	Fire Research and Management Exchange System
FSVeg	A USDA Forest Service database that contains point and plot vegetation data from field surveys such as Forest Inventory Assessment (FIA) exams, stand exams, forest inventories, and regeneration surveys. It includes data for trees, surface cover, understory vegetation, and downed woody material.
FVS	The USDA Forest Service's Forest Vegetation Simulator, a framework for modeling forest growth
FlamMap	A fire behavior mapping and analysis program that computes potential fire behavior characteristics (spread rate, flame length, fireline intensity, etc.) over an entire landscape for constant weather and fuel moisture conditions.
Fuels treatment	For the IFT-DSS, any mechanical, silvicultural, or burning activity whose main objective is to reduce fuel loadings or change fuel characteristics to lessen fire behavior or burn severity.
GDB	Geodatabase, a special form of geographic information system that can manage different types of geographic data
GIS	Geographic information system

TERM	DESCRIPTION
GUI	Graphical user interface
IFT-DSS	Interagency Fuels Treatment Decision Support System
IT	Information technology
JFSP	Joint Fire Science Program
LANDFIRE	Landscape Fire and Resource Management Planning Tools Project, a mapping project and database of vegetation, fire, and fuel characteristics
.lcp	Landscape File format
NEPA	National Environmental Policy Act
NEXUS	Crown fire hazard analysis software that links separate models of surface and crown fire behavior to compute indices of relative crown fire potential.
NIFCG	National Interagency Fuels Coordinating Group
NWCG	National Wildfire Coordinating Group
Navigator	The component that allows users to operate the system from a web browser
POC	Proof of concept
Project and Planning Database	A database for storing administrative data about fuels treatment projects
Pseudocode	An outline of step-by-step computer programming instructions that is not written in a particular programming language
RPC	Remote procedure call
SQL	Structured Query Language, used in managing databases
STS Study	Software Tools and Systems Study
Scenario Database	The component that stores representations of the scenarios that are run
Service Oriented Architecture (SOA)	An underlying structure that allows loosely connected components of a computer system to communicate, thus allowing services to be added or changed without creating a completely new structure
Software interface	The mechanism by which software components interact and communicate with one another.
Spatial Data Sequencer	The component that relays data between the Geodatabase and models
UML	Unified Modeling Language
WFDSS	Wildland Fire Decision Support System
WMS	Web Map Service, a protocol for (?)

EXECUTIVE SUMMARY

ES.1 INTRODUCTION

In May 2009, the Joint Fire Science Program (JFSP) initiated Phase IIIa of the Software Tools and Systems (STS) Study. Phase IIIa of the STS Study involves the development of a proof of concept (POC) system for the Interagency Fuels Treatment Decision Support System (IFT-DSS). The IFT-DSS POC will provide a user friendly software system to manage a subset of the most commonly used software tools and data to perform fuels planning scenarios. A key goal of the IFT-DSS POC is to demonstrate that a well-designed, extendable, service oriented architecture software (SOA) framework can help organize and manage the many existing data sets, software models, and tools in the fire and fuels domain and can help foster collaboration within a community of stakeholders. Ultimately, it is the goal of the IFT-DSS program to change the software development and deployment process within the fire and fuels domain to create efficiencies and leverage services among several large distributed SOA systems (e.g., the BlueSky Framework and the Wildland Fire Decision Support System, or WFDSS).

The IFT-DSS POC will explore the initial connections to two other SOA systems in the fire and fuels community: the BlueSky Framework and WFDSS. The BlueSky Framework has recently been expanded to provide access to its processes through web services. IFT-DSS will leverage BlueSky services where possible. The IFT-DSS development team will also work closely with the WFDSS development team to share technology. There is an overlap in functionality between the two systems, and IFT-DSS will reuse the work done by the WFDSS team to the maximum practical extent. In turn, as the IFT-DSS is developed, it will make its services available to the BlueSky Framework and WFDSS.

ES.2 THE FUELS TREATMENT PLANNING PROCESS

During Phase II and early in Phase IIIa, many efforts have been made to understand the decision support needs and the workflow processes involved in fuels treatment planning and management. As a result of these efforts, the following six workflow scenarios have been identified:

- •**Data acquisition and preparation** involves collecting and preparing the vegetation data needed for input into fire behavior and fire effects models.
- •**Strategic planning** involves identification of high fire hazard areas within an area of interest. The focus is to identify where further treatment analysis may be warranted on the basis of potential fire hazard.
- •**Spatially explicit fuels treatment assignment** involves (1) simulating fuels treatment placement in areas of high fire hazard within an area of interest; and (2) simulating post-treatment influences on fire behavior and fire effects potentials. The spatially explicit fuels treatment assignment extends the strategic planning analysis to applying treatments on the landscape.

- •**Fuels treatment effectiveness over time** involves the evaluation of the temporal durability of fuels treatments; that is, how long, in years to decades, a treatment will continue to affect potential fire behavior and fire effects within an area of interest. This workflow scenario naturally follows the strategic analysis and fuels treatment assignment workflow scenario.
- •**Prescribed burn planning** involves preparing the information needed to plan, document, and conduct a proposed prescribed fire.
- •**Risk assessment** involves conducting a probabilistic risk assessment for fuels treatment planning.

ES.3 IFT-DSS SOFTWARE ARCHITECTURE

The findings of the work performed during Phase I of the STS Study indicated that an SOA approach would best serve the fuels treatment community. Several key strategic-level requirements warrant a web-based, distributed SOA approach:

- The need for a system that can be easily accessed and used by fuels treatment specialists from a variety of different agencies.
- The need for a system that can organize, integrate, and manage existing and future fire and fuels software applications.
- The need for a system that will support distributed collaboration and allow fuels treatment planners to perform ad hoc analyses customized for a particular location and/or situation.

The core of the IFT-DSS software architecture consists of five elements: (1) a multi-layered graphical user interface, (2) a scenario database, (3) an executive application, (4) data and software services, and (5) a geospatial database. **Figure ES-1** illustrates the five key components of the architecture and their relative architectural arrangement. Each component contains interconnected sub-components, or layers, that perform specific functions within the system. These components and their behavior are described in Sections 3 and 4 of this document.

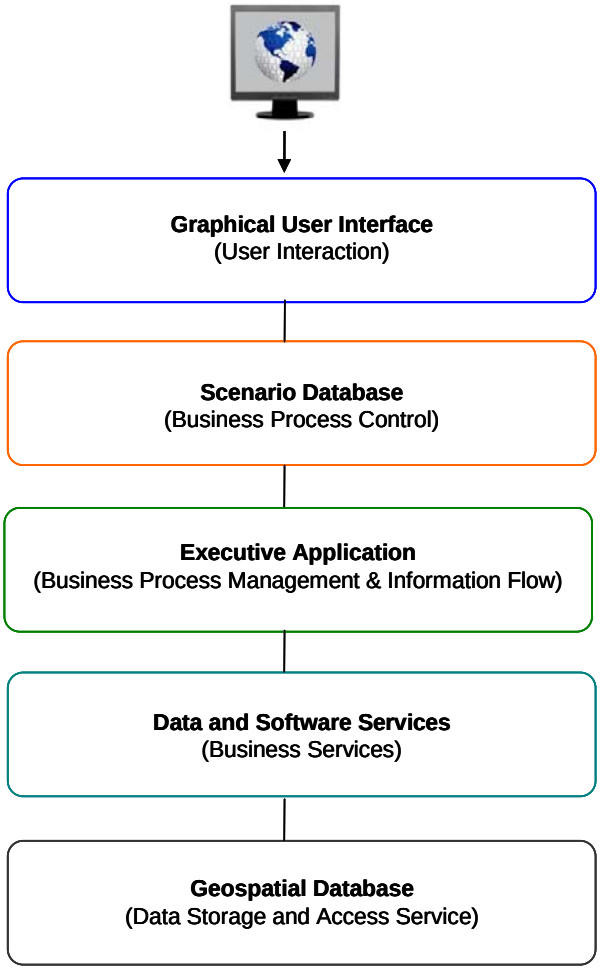


Figure ES-1. Illustration of the five key architecture components of the IFT-DSS.

The IFT-DSS architecture was designed to support the following key features:

- The IFT-DSS should make fuels treatment planning easier by
  - allowing users to acquire, create, and transform input data easily;
  - providing data choices: treelist, LANDFIRE grids, user supplied;
  - allowing users to view and edit spatial and tabular data (inputs and outputs);
  - organizing fuels treatment planning analysis steps and software tools; and
  - recognizing user error and explaining alternate action.
- The IFT-DSS should make fuels treatment planning more scientifically robust by
  - providing guidance regarding data and model choices based on the scale and type of analysis being performed;
  - allowing users to publish and share analysis methods and algorithms;
  - providing a mechanism to perform sensitivity and iterative analyses ;

- providing a mechanism to easily incorporate new models and tools as they are developed; and
- providing quality control, documentation, and audit-trail information to meet regulatory reporting requirements.

ES.4 IFT-DSS POC FUNCTIONALITY AND SCIENTIFIC MODELS

The IFT-DSS POC will implement three of the six workflow scenarios described in Section ES.2 above: 1) data acquisition and preparation, 2) strategic planning, and 3) prescribed burn planning. **Figure ES-2** lists the data, transformations, and scientific models that will be implemented in the IFT-DSS POC system.

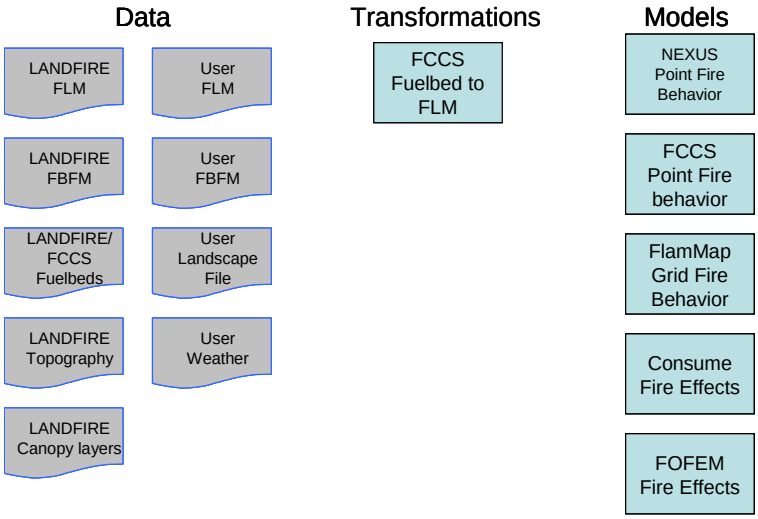


Figure ES-2. Data, transformations, and scientific models to be implemented in the IFT-DSS POC system.

ES.5 IFT-DSS USER EXPERIENCE

The IFT-DSS POC graphical user interface (GUI) will be web-based and will function within a standard web browser (for example, Internet Explorer or Firefox). From a user perspective, a project analysis will involve three phases: (1) project setup and planning; (2) software model execution and iterative analysis; and (3) project finalization, documentation, and archive. The IFT-DSS GUI will be designed based on early feedback from end users to ensure that it is straightforward and user friendly. **Figure ES-3** shows an example mock-up of the IFT-DSS GUI.

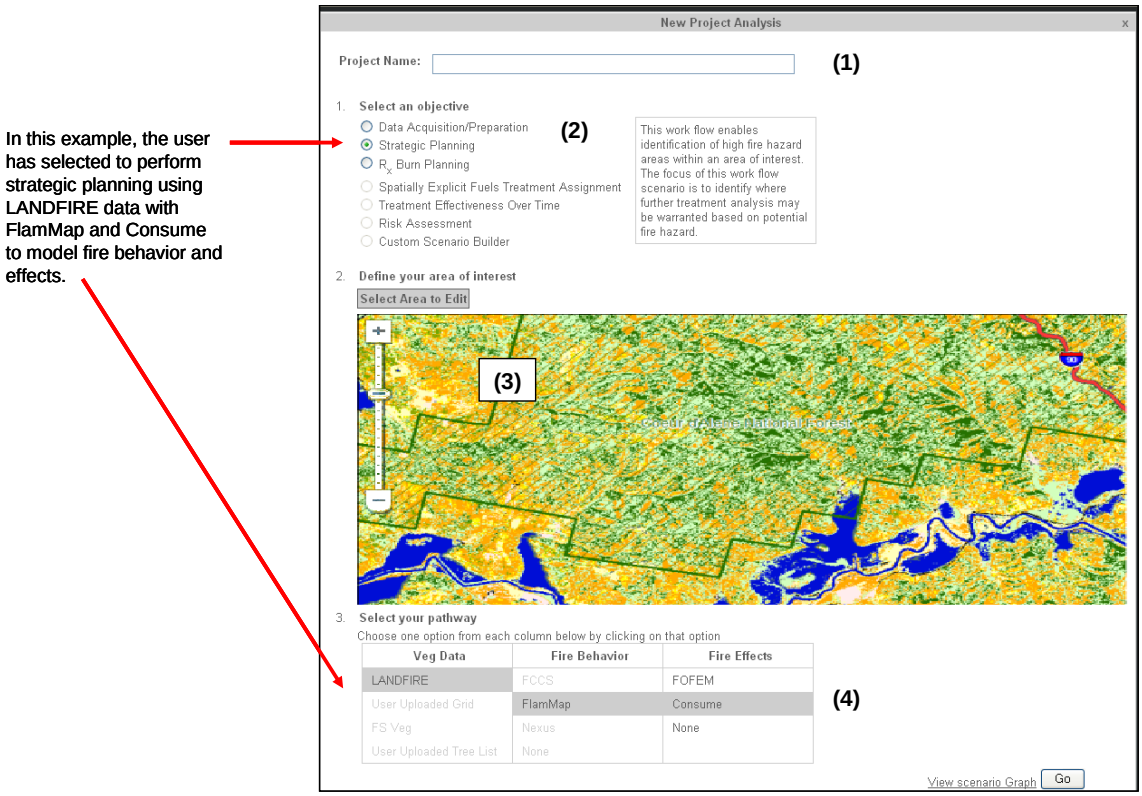


Figure ES-3. Mock-up of the IFT-DSS POC project setup screen. A user creating a new project analysis would go through four general steps: 1) provide a project name, 2) select an analysis objective, 3) define an area of interest; and 4) select a modeling scenario pathway.

ES.6 IFT-DSS SOFTWARE DESIGN

The IFT-DSS will serve as a software framework to integrate vegetation data, vegetation simulators, fire behavior and effects models, and risk analysis tools. It will support the reuse of IFT-DSS applications and services over the Internet, and it will be a flexible, modern, web-friendly system. In designing the IFT-DSS, both current and future needs of the user communities have been considered and therefore the IFT-DSS can be implemented as a general framework for scientific modeling and analysis. The IFT-DSS is designed in a manner that is generic and almost completely independent of programming language, hardware, and operating system implementation decisions; that is, the design can be realized in a variety of ways using any one of many hardware and operating system configurations. It is important to realize that this design was constructed in a way so that system components can be developed in different languages and use different operating foundations while still conforming to this design.

Figure ES-4 illustrates the overall system design including system components and interfaces, and their relationships. The IFT-DSS is described in terms of nine components and seven interfaces. The nine components are:

A. **Models** – The scientific and computational components of the system; all other components support the model operations

B. **Model Adaptors** – Enable integration of different models into the system

C. **Geodatabase (GDB)** – Stores the actual data inputted to and outputted from models

D. **Data Sequencer** – Relays data between the GDB and models

E. **Executive** – Directs the execution of the models

F. **Scenario Database** – Stores representations of how different "runs" of models are orchestrated

G. **Navigator Web Application** – Visualizes spatial data, allows users to edit data values for model inputs, and allows users to execute runs through scenarios or parts of scenarios, all from a web browser

H. **Project and Planning Database** – Stores administrative data about fuels treatment projects

I. **Planner Web Application** – Enables users to manage fuels treatment projects from a web browser

The seven interfaces are:

1. **Model Data Exchange** – Connections between machines running models to relay model inputs and outputs
2. **Model Control** – Connections between the Executive and model adaptors to prepare the model data exchange connections
3. **Sequencer Control and Monitoring** – Interaction between the Data Sequencer and Executive to manage scenario execution
4. **Executive Scenario Polling and Feedback** – Executive queries of the Scenario Database for required data, and feedback about model and machine execution performance
5. **Executive Launching and Progress Reporting** – Web application signaling of the Executive to compile and run a scenario segment and the Executive reporting progress back to the web application
6. **Map and Data Presentation Interface** – Communication between the GDB and the web application components
7. **Web Page Generator and Database Interface** – Web application queries of the Scenario Database to produce the web pages displayed to the user for model inputs and outputs

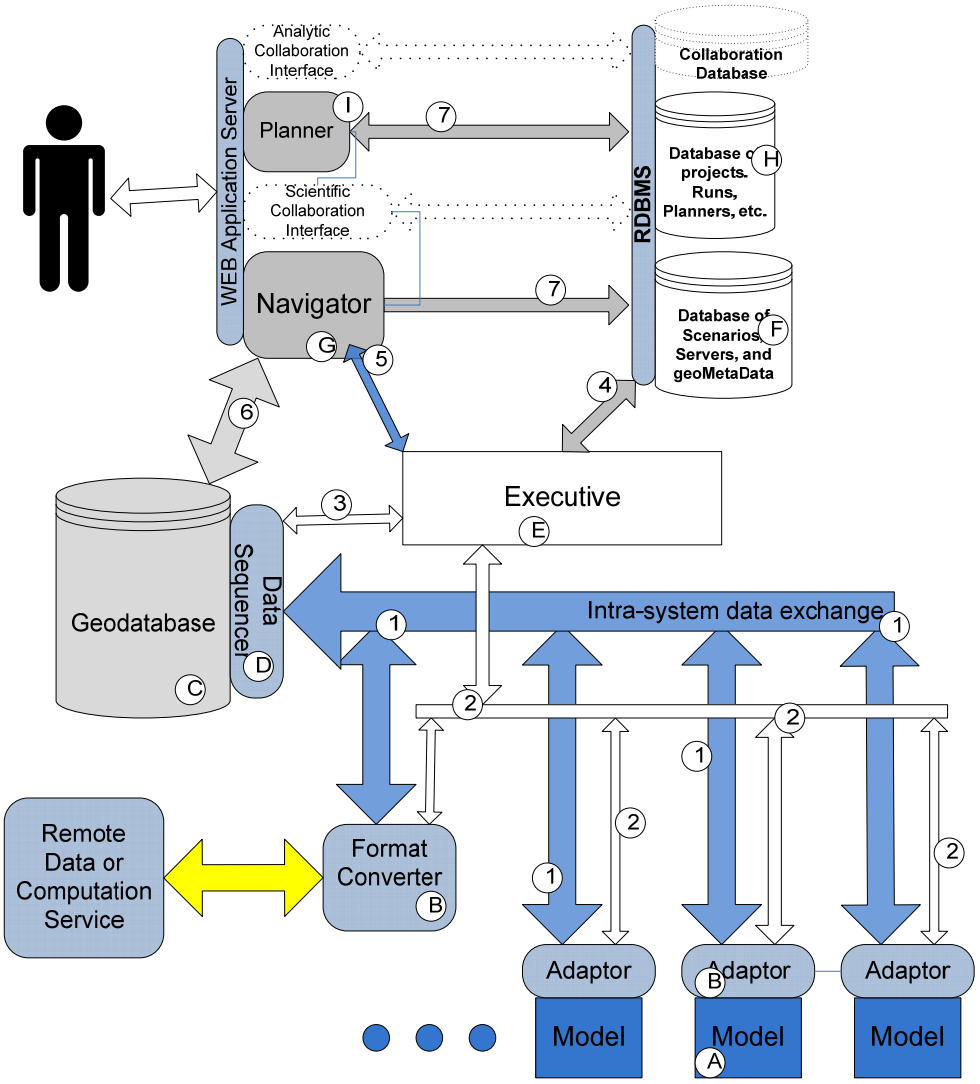


Figure ES-4. Overall structure of the IFT-DSS depicting the system's major components (with lettered indicators) and software interfaces (numbered). Features that will not be implemented in the IFT-DSS POC are represented with dashed borders

This design supports not only the immediate needs of the fuels treatment planning community but also supports two key requirements: 1) the ability for fuels treatment planners to collaborate and share analysis methods, and 2) the ability for the science and model development community to dynamically publish new data and applications to the system. In addition, this design is flexible and extensible to support future planning needs, new sources of data, and new applications as they evolve..

1. INTRODUCTION

In May 2009, the Joint Fire Science Program (JFSP) initiated Phase IIIa of the Software Tools and Systems (STS) Study. Phase IIIa of the STS Study involves the development of a proof of concept (POC) system for the Interagency Fuels Treatment Decision Support System (IFT-DSS). The IFT-DSS POC will provide a user friendly software system to manage a subset of the most commonly used software tools and data to perform fuels planning scenarios with the goal of demonstrating the usefulness and feasibility of the IFT-DSS. The development period for the IFT-DSS POC (Phase IIIa of the STS Study) will span approximately one year beginning in May 2009. This document contains the software design specifications for the IFT-DSS POC system.

The increasing operational complexity and urgency in fire and fuels management, coupled with a proliferation in the number of decision support tools available, have driven the need for a transformative solution. A distributed Service Oriented Architecture (SOA) approach that is readily accessible to users across different agencies has the potential to positively transform the development and deployment of data and software tools for fire and fuels management by organizing and coordinating the interaction of independently operating data services and software tools.[1] When developed with input and acceptance from all stakeholder communities, SOA systems can improve the exchange of information and promote collaboration among stakeholder communities. These systems can streamline the decision support process, facilitate improvements and advancements in fire and fuels science, and reduce the barriers that hinder the adoption of sophisticated risk management science concepts and practices.

Several SOA systems currently exist in the fuels treatment planning domain, and they are "distributed" and service-oriented to varying degrees. While these existing systems all contain useful science and link disparate software applications, some of the systems are inaccessible or require expert knowledge, and none of them individually supports the full range of fuels treatment planning activities. In addition, these systems have not been widely adopted; thus, their user groups are relatively small, and they have not been deployed in a way that facilitates interagency collaboration.

The IFT-DSS development effort is intended to demonstrate that a well-designed SOA approach, combined with community development efforts, can improve the decision support process and underlying science in the fire and fuels domain. The overall goal of the IFT-DSS is to develop an extendable software framework for organizing and managing the many existing data sets, software models, and tools available for fuels treatment planning and analysis and to foster collaboration within a community of stakeholders. Ultimately, it is the goal of the IFT-DSS program to change the software development and deployment process within the fire and fuels domain to create efficiencies and to leverage services among several large distributed SOA systems (e.g., the BlueSky Framework and the Wildland Fire Decision Support System).

[1] Refer to the Carnegie Mellon Software Engineering Institute (SEI) final report prepared for the JFSP for a definition and discussion about SOA, <http://frames.nbii.gov/documents/jfsp/sts_study/palmquist_sei_report_2008.pdf>

The IFT-DSS software architecture was designed to provide the following benefits to its various stakeholder communities:

- Integration, guidance, and collaboration regarding the use of existing data, software models, and tools for fuels treatment analysis and planning.
- Increased productivity and efficiency in the fuels treatment planning process through a system that can greatly reduce the time associated with preparing and manipulating data and software applications.
- A framework that facilitates peer review and model validation for scientific algorithms, applications, and vegetation data, along with a more efficient review, critique, and feedback mechanism to improve the scientific work flow and decision support process.
- A central framework that meets the needs of users, scientists and software developers, IT security specialists, and managers equally well, organizing a myriad of software systems in a functionally effective, user-friendly manner while allowing IT administrators to provide appropriate security and access.
- Another vehicle to enhance interagency functionality and collaboration and to serve as a proving ground for identifying and testing acceptable governance issues that best support interagency operations.

To achieve these goals, an SOA approach was taken in designing the IFT-DSS architecture. 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 (distributed) so that users can combine and reuse individual services as needed. A key characteristic of a distributed SOA is the ability of users to have choices and control over the data and software applications that are applied to address a specific situation.[2]

In the coming months, Sonoma Technology, Inc. (STI) will develop a guidance document that will include information about how new software applications and services can be made accessible to the IFT-DSS, how other SOA systems in the fire and fuels community can share services with the IFT-DSS and vice versa, and how the system can be maintained at a future host agency.

1.1 PURPOSE

This document is intended to serve two purposes: (1) to provide background and an overview of the IFT-DSS POC system from a user perspective, and (2) to provide technical design specifications and a plan for the software implementation of the POC system through May 2010. Sections 1, 2, and 3 of this document are intended for the end user of the IFT-DSS, with Sections 1 and 2 providing an introduction to and background of the STS Study and the purpose and functionality of the IFT-DSS, and Section 3 containing a discussion of the graphical user interface (GUI) for the POC system. Sections 4 and 5 are primarily for the use of the IFT-DSS software development team and contain the technical design specifications for the IFT-DSS POC.

[2] Palmquist, 2008 (pages 17-18), <http://frames.nbii.gov/documents/jfsp/sts_study/palmquist_sei_report_2008.pdf>.

There is no standard approach or content requirement for software design specification documents; however, in general, a software design specification should provide a technical implementation plan for the development team (and new team members) as well as objective evidence that the team will develop a system that meets the system requirements. As such, this software design specification provides enough design detail to serve as a guide but does not include so much detail as to hinder a flexible, or agile, development process.

This document provides sufficient detail to ultimately serve as the testing and quality assurance (QA) guidance document used by the IFT-DSS testing and QA team to ensure that the POC system meets the functions and specifications contained herein. At the same time, this document will be updated throughout the IFT-DSS POC development process to reflect design changes that may occur. It will be considered a living document and will eventually serve as the final technical documentation for the POC system.

1.2 DOCUMENT ORGANIZATION

This document is organized into six main sections.

- **Section 1 – Introduction**: Provides background on the IFT-DSS and the purpose of this document.
- **Section 2 – System Overview**: Provides an overview on the user community, anticipated work flows, and the system architecture.
- **Section 3 – The IFT-DSS POC User Experience**: Provides examples and a discussion of what the IFT-DSS POC might look like to users of the system.
- **Section 4 – Technical Software Design**: Provides a high-level description of the IFT-DSS design.
- **Section 5 – Technologies**: Provides a discussion of technologies that are expected to be used in the initial development of the POC.
- **Section 6 – References**: Provides a list of references cited in the document.

In addition to these sections, this document includes an executive summary and a glossary. Because this document, and especially Section 4, contains many technical terms, the reader may find the glossary particularly helpful. The glossary is located near the beginning of this document just before the executive summary.

1.3 IFT-DSS NAMING CONVENTION

There are likely to be several versions of the IFT-DSS as development progresses. **Figure 1-1** illustrates and describes the naming convention for the IFT-DSS.

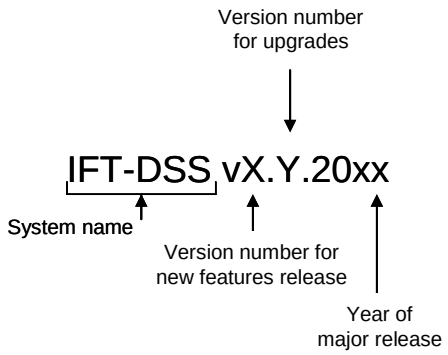


Figure 1-1. Naming convention for the IFT-DSS.

Table 1-1 defines the tentative IFT-DSS naming convention proposed for the next three years of development. Note that there may be interim versions of the IFT-DSS that contain upgrades to the versions indicated in Table 1-1.

Table 1-1. IFT-DSS proposed naming convention during development.

IFT-DSS Version	Description
IFT-DSS v1.0.2010	The POC system described in this document that will be released in May 2010.
IFT-DSS v2.0.2011	Version 2 of the system to be released in May 2011.
IFT-DSS v3.0.2012	Version 3 of the system to be released in May 2012

1.4 REFERENCES TO OTHER RELEVANT DOCUMENTS

Several documents have been produced throughout the STS Study that are relevant to this document. All STS Study documentation can be accessed through the Fire Research and Management Exchange System (FRAMES) website. The following documents are of particular relevance to this document:

- Working Summary of the SEI's Engagement with the Joint Fire Science Program (Palmquist, 2008).
- The Interagency Fuels Treatment Decision Support System Software Architecture (Funk et al., 2009),
- Refined Work Flow Scenarios for the Interagency Fuels Treatment Decision Support System (Drury et al., 2009).

2. SYSTEM OVERVIEW

Any number of architectural approaches could have been proposed for the IFT-DSS; however, there are key requirements that the IFT-DSS must meet that warrant a web-based, distributed SOA approach:

- The need for a system that can be easily accessed and used by fuels treatment specialists from a variety of different agencies.
- The need for a system that can organize, integrate, and manage existing and future fire and fuels software applications.
- The need for a system that will support distributed collaboration and allow fuels treatment planners to perform analyses customized for a particular location and/or situation.

This section provides a strategic-level overview of the fuels treatment decision support process and the IFT-DSS architecture, including the rationale for the design approach and a discussion of the key components of the system. A complete list of system requirements for the IFT-DSS was developed as part of the architecture design process (Phase II of the STS Study) and can be found in the document entitled *The Interagency Fuels Treatment Decision Support System Software Architecture* (Funk et al., 2009),

2.1 SERVICE ORIENTED ARCHITECTURE

The term SOA is not an alias for a particular system; rather, it describes a particular way in which a system is constructed. For our purpose, SOA is defined as a generic software architecture framework designed to support a collection of services (databases and software applications) with well-defined software interfaces.[3] SOA facilitates the integration of new and legacy software applications to streamline work processes. This architectural approach can also support inter-operability with other decision support systems in the fire and fuels domain such as the BlueSky Framework and the Wildland Fire Decision Support System (WFDSS). A "distributed" SOA is a system whose components are (or can be) distributed across a computer network; that is, the services within the system can reside in geographically different locations and can be accessed across a network. A web-based, distributed SOA is a system that is accessible to users and controlled through a standard web browser (e.g., Internet Explorer, Firefox, etc.) interface.

[3] A software interface is the programmatic mechanism that allows components (software applications) to communicate with one another so that data and services can be accessible and interoperable within an SOA framework.

2.2 RATIONALE FOR A SERVICE ORIENTED ARCHITECTURE APPROACH

The findings of the work performed during Phase I of the STS Study indicated that an SOA approach would best serve the fuels treatment community. Several key strategic-level requirements warrant a web-based, distributed SOA approach:

- **The need for a system that can be easily accessed and used by fuels treatment specialists from a variety of different agencies.** Government and state agencies often have rules and regulations regarding installation of new software on government computers and workstations. These regulations make it difficult to install software applications directly onto agency desktop computers. Therefore, to make the IFT-DSS accessible across agencies, the system must be able to run on a standard desktop (or laptop) computer with an Internet connection—eliminating the need for users to install new software on their local computer. This feature requires that the system be hosted on an accessible server and developed to function within the most commonly used web browsers (e.g., Internet Explorer, Firefox).
- **The need for a system that can organize, integrate, and manage existing and future fire and fuels software applications.** Existing fire and fuels software applications perform a variety of functions and can be combined to perform complex simulations. The variety of existing applications presents an integration challenge, as these applications were developed with different goals and requirements, at different times, and by different organizations. In addition, these applications were written in different software languages, may currently run on different operating systems, have overlapping functionality, and require differing data formats. The SOA approach evolved to address the issue of integrating such disparate services and software applications. The design of the IFT-DSS software architecture is driven by the requirement to enable disparate applications to work together (including applications not yet developed) while requiring minimal additional effort from application developers to support the framework. This objective can be achieved by designing an SOA architecture that is adaptable and generic enough to accommodate a broad variety of applications and functionality.
- **The need for a system that will support distributed collaboration and allow fuels treatment planners to perform ad hoc analyses customized for a particular location and/or situation.** One of the findings from Phase I of the STS Study was that fuels management and risk mitigation require a distributed approach to collaboration and present an ongoing need for data fusion. Because of the variety of operational contexts, it is impossible to know the exact sets of models, tools, or data sets needed for every fuels treatment planning situation. Therefore, customizable approaches are needed, requiring collaborative tools that support web-enabled methods of analysis. A flexible and extendable software framework will allow tool developers or sophisticated users to rapidly configure, calibrate, or extend web-enabled capabilities to meet needs of a specific operational situation.

2.3 IFT-DSS USERS AND STAKEHOLDERS

A major goal of the overall IFT-DSS is to develop a community of individuals from multiple agencies and organizations that can collaborate, exchange, and communicate science and information related to fuels treatment analysis and planning. Collaboration among the fire and fuels community is important to improve the science and understanding of fuels treatment planning and to keep the data, software applications, and the IFT-DSS updated to meet current and future needs. In addition, collaboration helps all agencies and organizations learn from each other about methods, challenges, and approaches for fuels treatment planning.

Through the efforts of Phase II of the STS Study, five stakeholder groups were identified for the IFT-DSS:

1. Approximately 1,000 **fire and fuel operations managers,** or fuels treatment specialists, at multiple federal and state agencies throughout the United States. Fuels treatment specialists will be the primary users of the system for year-round fuels treatment planning.
2. Several (20) **scientist developers** who will provide new or updated science, models, and tools to the system. Scientific collaborators will be periodic contributors to and users of the IFT-DSS.
3. **Database developers** who will provide applications and data to the system, including:
   - a few (2 to 5) institutional application and data providers who will make large-scale databases available to the system (e.g., LANDFIRE and FSVeg); and
   - fuels treatment specialists who will upload or create local data sets.
4. **Information technologists and software specialists** who will operate and maintain the system over time.
5. **Agency senior management,** including the National Wildfire Coordinating Group (NWCG), who are responsible for issues related to business needs, resource allocation and prioritization, financial investments, and the operational efficiency and effectiveness of the community as a whole.

A community development effort to engage each of the five stakeholder groups listed above is being conducted in parallel with the IFT-DSS POC software development effort. The purpose of the community development effort is to gain the support and acceptance of a network of stakeholders that gain valuable services from the IFT-DSS and have responsibilities for the on-going functional and maintenance aspects of the system. In particular, the community development strategy will (a) describe the various stakeholder communities and their characteristics; (b) present a plan for enhancing awareness about and use of the IFT-DSS software by various subgroups of these stakeholder communities; and (c) provide a roadmap of

how the IFT-DSS development team proposes to transition the software from the originators, JFSP and the NIFCG, to an agency Managing Partner for the NWCG.[4]

2.4 OVERVIEW OF THE FUELS TREATMENT DECISION SUPPORT PROCESS

At the most basic level, the fuels treatment analysis and planning process involves performing environmental assessments of fuels treatment options as mandated by the National Environmental Policy Act (NEPA). This decision process centers on managing outcomes by modifying vegetation. The decision support process involves preparing a detailed vegetation data set; modeling vegetation changes based on growth, treatments, and/or disturbance; and analyzing the results of the modeled vegetation. A fuels treatment specialist then recommends which treatment option to apply.

During the past year, many efforts have been made to understand decision support needs and the work flow processes involved in fuels treatment planning and management. These efforts included surveying the fuels treatment planning community; conducting personal interviews with several fuels treatment specialists representing different land management agencies; engaging and soliciting feedback from the Interagency Fuels Treatment Work Group (IFTWG); and conducting meetings and discussions with fire and fuels software application and data developers. As a result of these efforts, the following six work flow scenarios have been identified:

- **Data acquisition and preparation work flow scenario** provides a simple and efficient way to collect and prepare the vegetation data needed for input to fire behavior and fire effects models.
- **Strategic planning work flow scenario** enables identification of high fire hazard areas within an area of interest. The focus of this work flow scenario is to identify where further treatment analysis may be warranted on the basis of potential fire hazard.
- **Spatially explicit fuels treatment assignment work flow scenario** (1) simulates fuels treatment placement in areas of high fire hazard within an area of interest, and (2) simulates post-treatment influences on fire behavior and fire effects potentials. The spatially explicit fuels treatment assignment work flow scenario extends the strategic planning analysis to applying treatments on the landscape.
- **Fuels treatment effectiveness over time work flow scenario** enables the evaluation of the temporal durability of fuels treatments, that is, how long, in years to decades, a treatment will continue to lower potential fire behavior and fire effects within an area of interest. This work flow scenario naturally follows the strategic analysis and fuels treatment assignment work flow scenario.
- **Prescribed burn planning work flow scenario** provides the information needed to plan, document, and conduct a proposed prescribed fire.

[4] For more specific information regarding the community development plan, refer to the FRAMES website at http://frames.nbii.gov.

- **A proposed risk assessment work flow scenario** provides a probabilistic risk assessment for fuels treatment planning.

The work flow scenarios presented here can be divided into two categories: (1) fuels treatment work flow scenarios and (2) a prescribed burn planning work flow scenario. The prescribed burn planning scenario is often considered one phase of a fuels treatment scenario because prescribed burning is one way to treat fuels. However, the development of a prescribed burn plan is a long and complex process, and fuels treatment specialists have indicated that the IFT-DSS could provide a useful service by supporting a work flow scenario specifically devoted to prescribed burn planning.

The work flow scenarios defined here are not mutually exclusive and generally build on one another. The IFT-DSS architecture is designed to be flexible and scalable so that new work flow scenarios can be implemented as they evolve. **Figure 2-1** is a work flow diagram showing how the six work flow scenarios fit together from a process perspective. The IFT-DSS POC will implement a subset of the work flow scenarios described in this section (see Section 2.6.2).

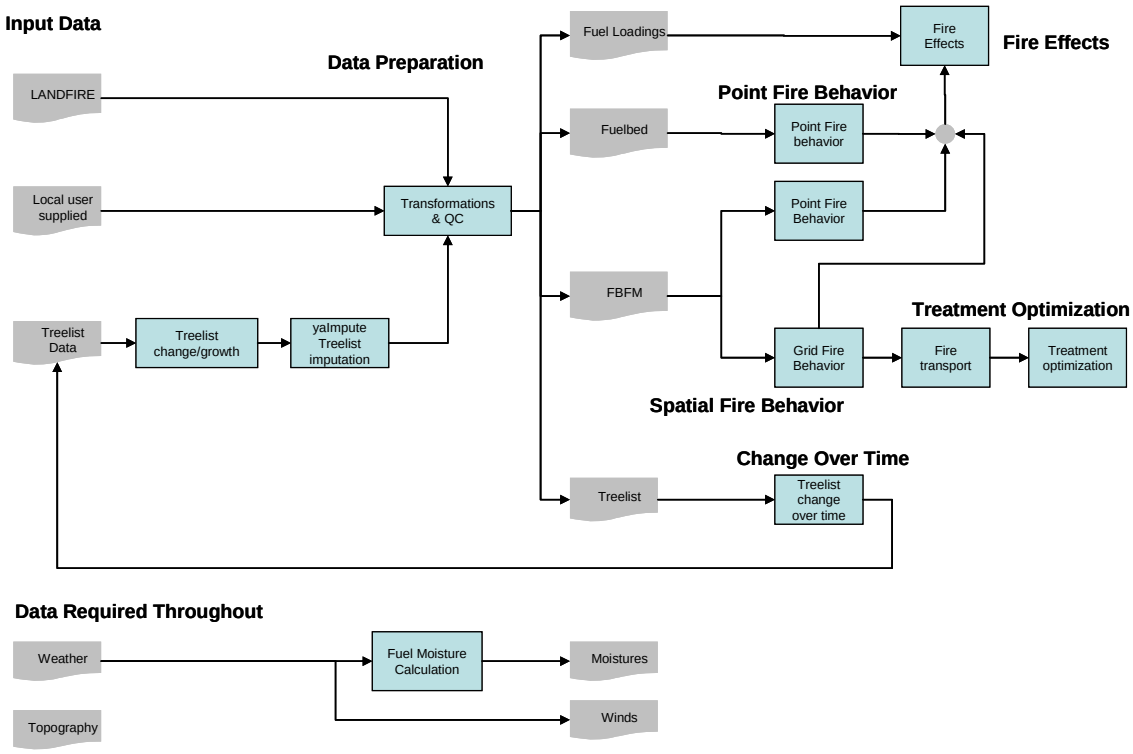


Figure 2-1. Work flow diagram illustrating how the fuels treatment planning work flow scenarios fit together from a process perspective.

2.5 IFT-DSS POC REQUIREMENTS

Eight general requirements were established for the IFT-DSS. These requirements and a brief description of how they will be realized in the POC are summarized in **Table 2-1**.

Table 2-1. IFT-DSS requirements and how they will be realized in the POC.

Requirement	POC Realization
1. Support the decision support process, analysis steps, and software tools commonly used for fuels treatment planning.	Implement three of the most common work flow scenarios: 1. Data acquisition and preparation 2. Strategic planning 3. Prescribed burn planning.
2. Provide users with software model choices within work flow scenarios.	Implement NEXUS, FCCS, and FlamMap as choices for fire behavior and effects modeling.
3. Provide data choices.	Implement three data input options: 1. User-supplied local data in .lcp file format 2. User-supplied manual entry of data for single unit fire behavior modeling 3. LANDFIRE data as default
4. Support visualization of spatial and tabular data, data editing, and user interaction at each processing step.	Implement interactive GIS map viewer and editing tool.
5. Have a quality control, documentation, and audit-trail mechanism to support regulatory requirements.	Initially implement reporting tools that will support the first three work flow scenarios. At a minimum, the system will generate maps, graphs, and data tables, as well as information that can be easily used in a prescribed burn plan.
6. Support analytical collaboration; that is, the system should provide a mechanism for fuels treatment analysts to publish and share methods and algorithms with other system users via a central system library.	Users will have the ability to publish their methods and results for other users to view.
7. Support scientific collaboration; that is, the system must be able to incorporate new models and tools as they become available through an authorship and publishing mechanism.	Initially implement three fire and fuels applications: NEXUS, FCCS, and FlamMap. Develop software guidelines for integrating new models into the system. See discussion of model integration approach in Section 2.8.
8. Provide an easily accessible and straightforward user interface to all models and applications.	Implement a web-based, easy-to-use GUI.

2.6 OVERVIEW OF THE IFT-DSS ARCHITECTURE

This section is a strategic-level overview of the IFT-DSS software architecture. A complete description of the architecture is provided in the IFT-DSS Software Architecture Design Document[5].

The IFT-DSS will serve as a software framework to integrate disparate vegetation data, vegetation simulators, fire behavior and effects models, and risk analysis tools (some of which are web-based) into a common GUI. It will support the reuse of IFT-DSS applications and services over the Internet, and it will be a flexible, modern, web-friendly system.

2.6.1 Architectural Components

The core of the IFT-DSS software architecture consists of five elements: (1) a multi-layered GUI, (2) a Scenario Database, (3) an executive application, (4) data and software services, and (5) a geospatial database. A description of the general function(s) of these five key architecture components follows:

- The multi-layered graphical user interface (GUI)—Provides the user with an access point into the IFT-DSS. It controls the user experience and is the single portal for all system inputs and outputs.
- The Scenario Database—Stores all information, including data and user information that is input to the IFT-DSS via the GUI. The Scenario Database manages the user experience and the decision support process.
- The Executive application—Works with the Scenario Database to invoke specific functions and services. It functions as the internal system process controller and manages the flow of data and information within the system.
- The data and software services—Provide the fire and fuels treatment domain data, software models, and tools that support the decision support process.
- The geospatial database—Stores all project-related input, intermediate, and output data. It is called the geospatial database because it contains spatial data (map layers) but it also can contain non-spatial tabular data.

Figure 2-2 illustrates the five key components of the architecture and their relative architectural arrangement. Each component contains interconnected subcomponents, or layers, that perform specific functions within the system. These components and their behavior are described in Sections 3 and 4.

[5] http://frames.nbii.gov/documents/jfsp/sts_study/ift_dss_task2_tech_architecture_draft_20090212.pdf

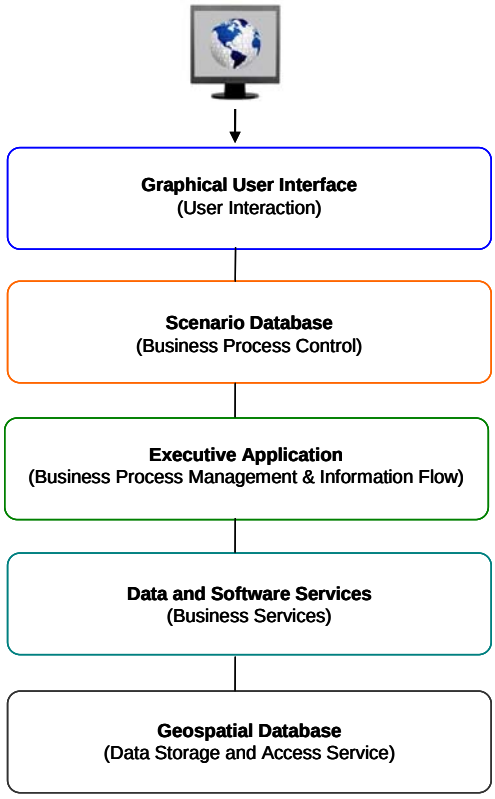


Figure 2-2. Illustration of the five key architecture components of the IFT-DSS.

2.6.2 Key Architectural Features and Functions

The IFT-DSS architecture is designed to support the following functions and features:

- organize the decision support process, analysis steps, and software tools commonly used for fuels treatment planning;
- provide data choices (i.e., standard treelist data, standard gridded data, and/or locally generated data);
- enable visualization of spatial and tabular data, editing of data, and user interaction at each processing step;
- streamline data preparation and processing by offering a mechanism to acquire, create, and transform input data (e.g., the ability to combine vector and raster data formats, perform vector-to-raster transformations, and vice versa);
- provide a quality control, documentation, and audit-trail mechanism to meet regulatory reporting requirements;
- provide guidance on user options (e.g., submodel choices) based on geographic scale and the type of analysis being performed;
- enable the stopping and starting of analyses at any processing point;

- facilitate analytical collaboration through a central system library that allows fuels treatment analysts and scientists to publish and share methods and algorithms with other system users;
- provide a mechanism to perform sensitivity analyses;
- recognize user error and explain alternate actions; and
- facilitate and encourage scientific collaboration through an authorship and publishing mechanism that is able to incorporate new models and tools as they become available.

2.7 IFT-DSS POC FUNCTIONALITY

The IFT-DSS POC will implement a subset of the work flow scenarios and functions described in Section 2.3. The goal for the IFT-DSS POC at the end of the first year of development (May 2010) is to deliver a proof of concept product that is useful for the fuels treatment planning community. To maximize the functional capabilities of the POC within the limited, one-year time frame, we have chosen to focus on the implementation of the data acquisition and preparation, strategic planning, and prescribed burn planning work flow scenarios.

2.7.1 Data and Model Implementation

The software applications and data listed in **Figure 2-3** have been chosen for initial implementation because (1) they have been identified as widely adopted and used by the fire and fuels community; and (2) they are likely to be the most technologically feasible to incorporate into the system during the first year of development.

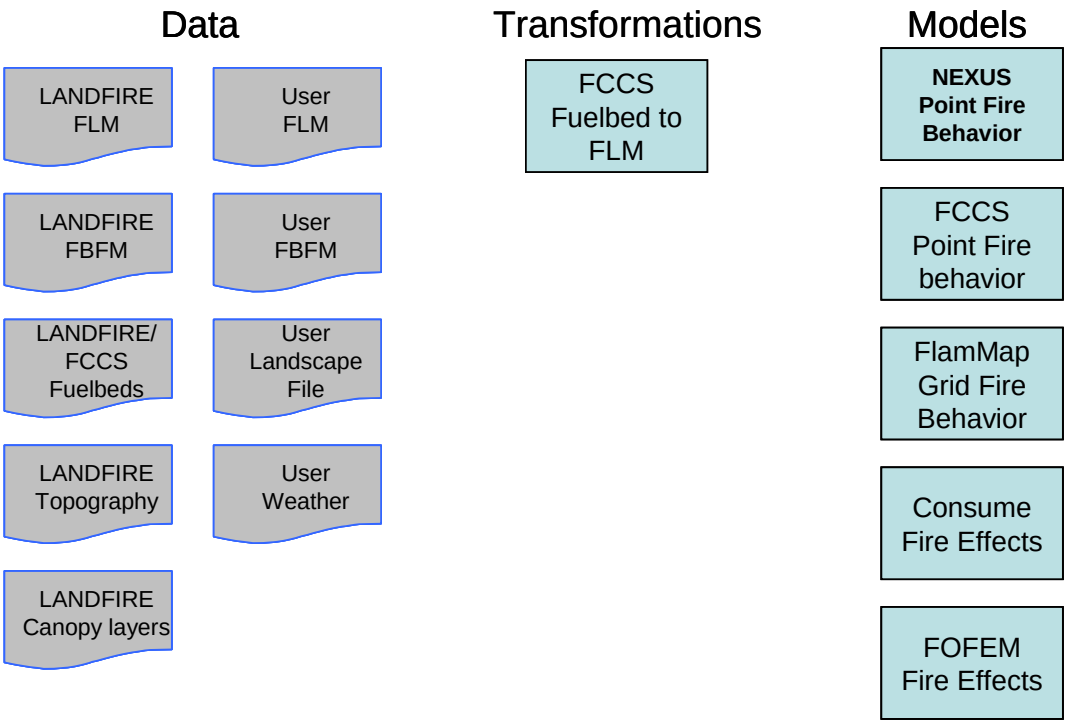


Figure 2-3. Data, transformations, and models to be implemented in the IFT-DSS POC system.

2.7.2 IFT-DSS POC Work Flow Scenarios

This section describes the three fuels treatment planning work flow scenarios that will be implemented in the IFT-DSS POC: 1) data acquisition and preparation, 2) strategic planning, and 3) prescribed burn planning.

Data Acquisition and Preparation

The objective of the data acquisition and preparation work flow scenario is to acquire, prepare, and quality assure vegetation data for use in fuels treatment planning. In the IFT-DSS POC, the data acquisition functionality will provide the user with three options: (1) to manually enter data needed for prescribed burn planning (i.e., fuel model, fuel loadings, wind speed, etc.) into a form via the user interface; (2) to upload local data in Landscape File (.lcp) format; and (3) to access LANDFIRE National data. The option to manually enter data will be implemented first, followed by the option to access LANDFIRE National data and by the import functions to upload local .lcp files. As the POC is developed, other functionality, such as the ability to edit .lcp files and LANDFIRE data for local conditions, will be added. **Figure 2-4** illustrates the work flow process for data acquisition and preparation as it will be implemented in the IFT-DSS POC.

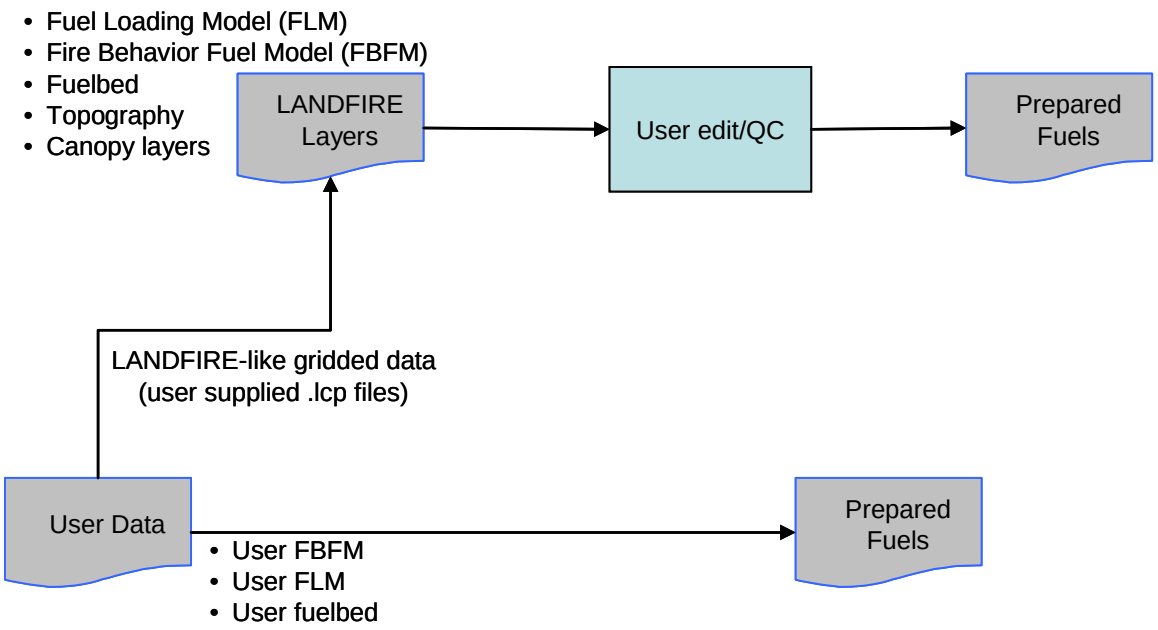


Figure 2-4. Process diagram for the data acquisition and preparation work flow scenario to be implemented in the IFT-DSS POC.

Strategic Planning Work Flow Scenario

The objective of the strategic planning work flow scenario is to identify high fire hazard areas within an area of interest. The focus of this work flow scenario is to identify locations where the potential fire hazard may warrant further analysis. High fire hazard is expressed by high potential fire behavior and/or undesirable fire effects.

In the IFT-DSS POC, FlamMap will be implemented to perform fire behavior simulations across all pixels in an area of interest. Once FlamMap is functional, work will begin to implement the fire effects models Consume and FOFEM. Consume will be implemented first, followed by FOFEM.

FlamMap fire behavior outputs include fireline intensity, flame length, rate of spread, heat per unit area, horizontal movement rate, midflame wind speed, spread vectors, and crown fire activity. Consume fire effects outputs are fuel consumption, smoke emissions, and heat release, while FOFEM simulates tree mortality, fuel consumption, smoke emissions, and soil heating. **Figure 2-5** summarizes the strategic analysis work flow scenario as it will be implemented for the POC.

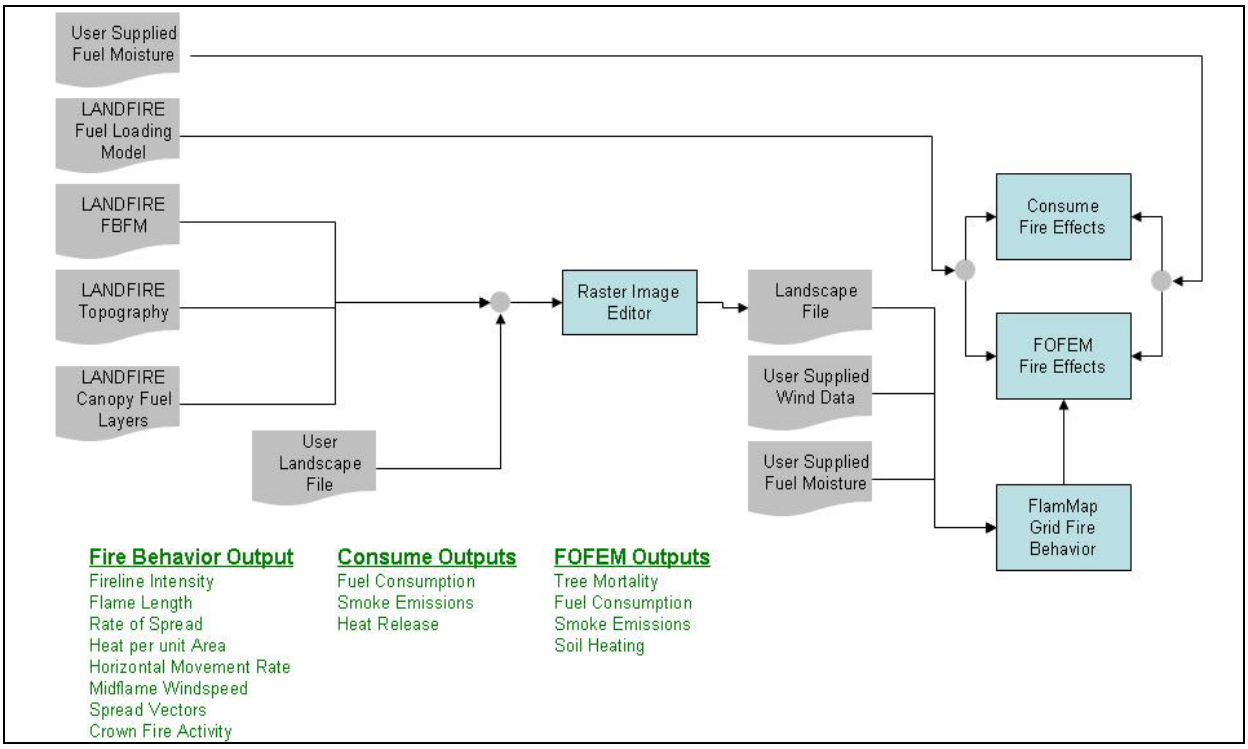


Figure 2-5. Process diagram for the strategic planning workflow scenario as it will be implemented in the IFT-DSS POC.

Prescribed Burn Planning Work Flow Scenario

The objective of the prescribed burn planning work flow scenario is to provide the information needed to plan, document, and conduct a proposed prescribed fire. The IFT-DSS POC will have two pathways for fire behavior simulations: running FlamMap for a single point location and using the Fuel Characteristic Classification System (FCCS) fire behavior calculator. For both options, users will manually input data parameters such as wind, fuel models, fuel moisture, and fuel loadings. If a user selects the FlamMap pathway, FlamMap will function behind the scenes on a point location. In the FCCS pathway, the FCCS fire behavior calculator will function on individual fuelbeds. Fire effects will be simulated by either Consume or FOFEM. Simulations of possible fire behaviors provide essential information for describing the burn plan prescription (Element 7), the ignition plan (Element 15) and the holding plan (Element 16).[6]

FlamMap Pathway

Using the FlamMap pathway in the POC, the user will input data parameters required by FlamMap. As described in the strategic planning scenario, FlamMap's outputs include fireline intensity, flame length, rate of spread, heat per unit area, horizontal movement rate, midflame

[6] U. S. Department of Agriculture and U. S. Department of the Interior (2008) Interagency prescribed fire planning and implementation procedures guide. Available on the Internet at <http://www.nifc.gov/fire_policy/rx/rxfireguide.pdf>.

wind speed, spread vectors, and crown fire activity. This information is crucial in determining how and when a prescribed burn should be conducted to meet specific objectives. Consume and FOFEM will provide information about fire effects.

In addition to allowing direct user input of fuels data, the POC will allow users to use the LANDFIRE National data layers for this workflow scenario. When the user specifies the geographic coordinates for the prescribed burn, the appropriate data will be retrieved, and the fire behavior and fire effects programs will run simulations for the specified location. **Figure 2-6** summarizes the process steps in this pathway.

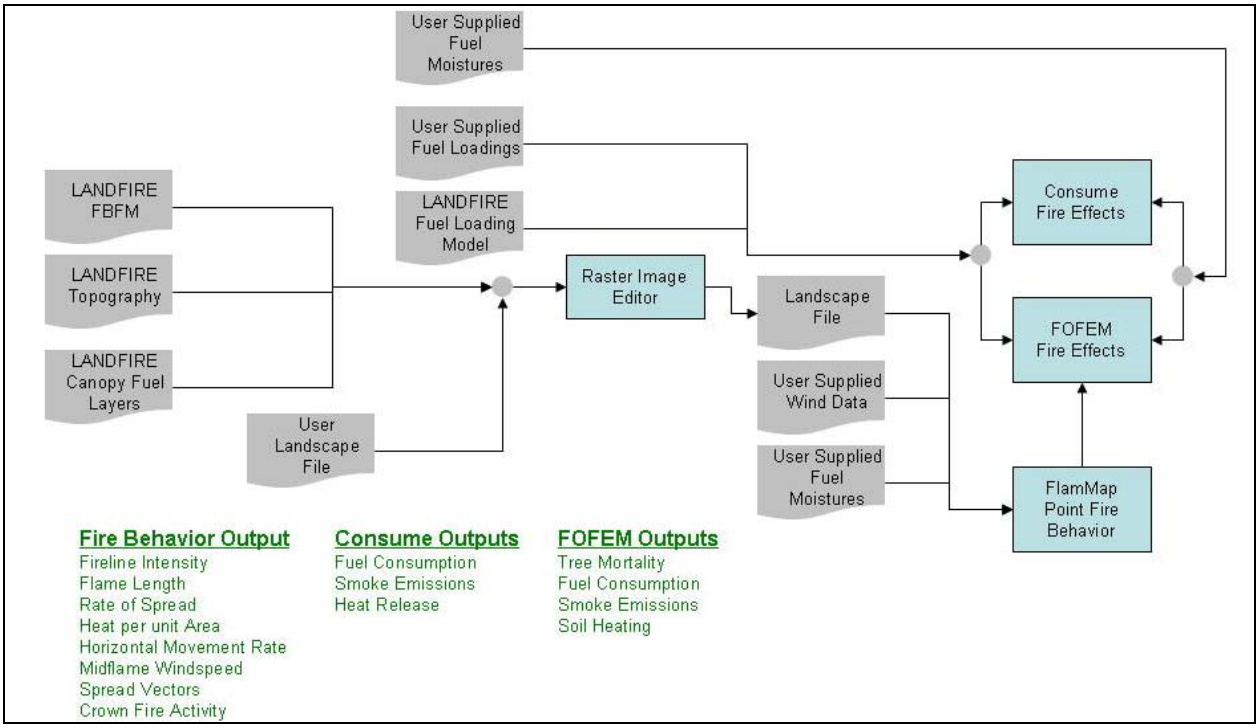


Figure 2-6. Process diagram for the prescribed burn planning work flow scenario using the FlamMap pathway in the IFT-DSS POC.

Fuel Characteristic Classification System Pathway

For the FCCS pathway, the POC will provide the surface and crown fire behavior outputs currently found in the desktop version of the FCCS. The most useful outputs for prescribed burn planning are reaction intensity, flame length, rate of spread, and crown fire potential. **Figure 2-7** summarizes the FCCS pathway for the prescribed burn planning work flow scenario and provides more information about the data inputs and the fire behavior outputs from the FCCS calculator.

In the POC, users will have the ability to manually enter fuelbeds into the FCCS, in much the same way as they do with the current desktop version. In addition, a default set of gridded fuelbeds will be provided. As functionality increases, users will have the ability to manually edit

fuelbeds within the FCCS, using an FCCS fuelbed editor similar to that available in the standalone version of the FCCS; however, this functionality may not be available in the POC at the end of the first year of development.

The FCCS pathway will eventually include linkages to Consume and FOFEM. The Consume outputs (fuel consumption, smoke emissions, and heat released) and the FOFEM outputs (tree mortality, fuel consumption, smoke emissions, soil heating) provide additional information to help planners address how to conduct a prescribed burn to meet the objectives described in the burn plan.

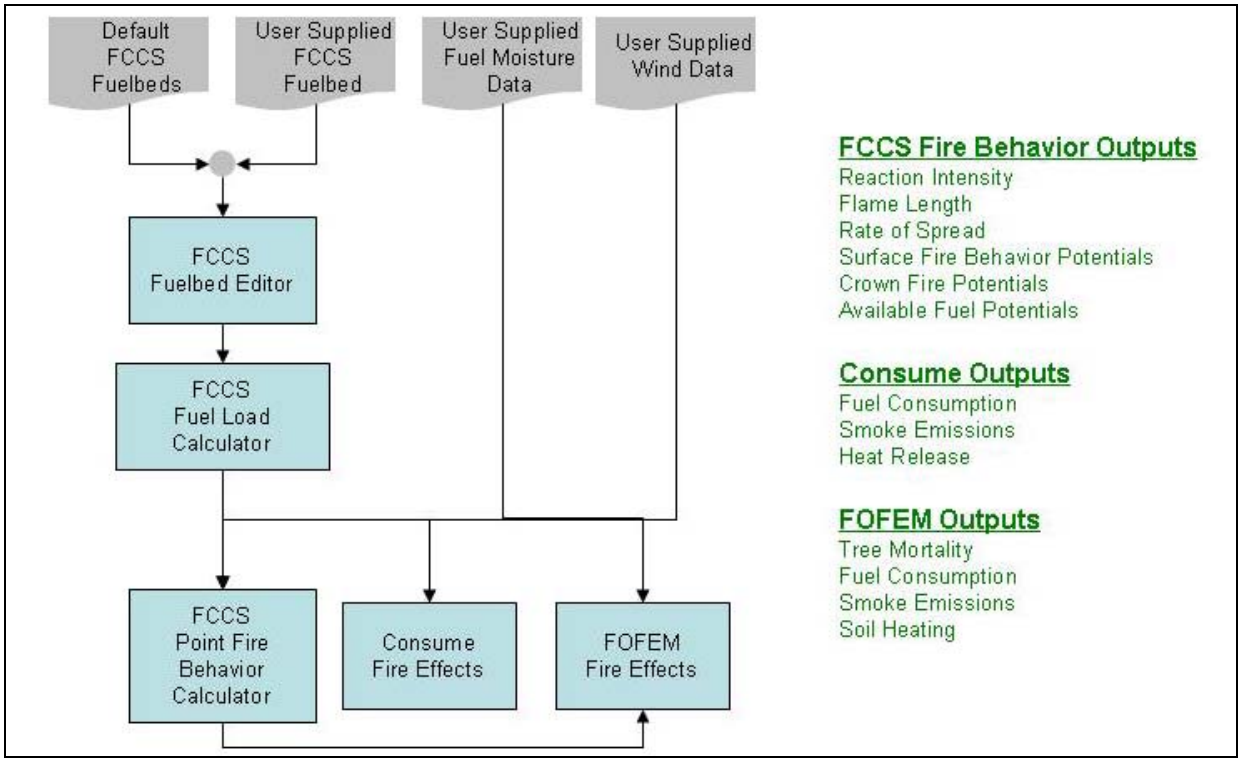


Figure 2-7. Process diagram for the prescribed burn planning work flow scenario using the FCCS pathway in the IFT-DSS POC.

NEXUS Pathway

Using the NEXUS pathway in the POC, the user will manually input data parameters required by NEXUS. NEXUS outputs will include fire type, crown fraction burned, flame length, rate of spread, heat release, fireline intensity, mid-flame wind speed, crowning index, torching index, surfacing index, critical fireline intensity, critical flame length, critical canopy bulk density, and critical spread rate. Note that some of the outputs of NEXUS are different from those of FlamMap and FCCS.

2.7.3 IFT-DSS POC Implementation

Figure 2-8 illustrates the pathways (colored lines) and the order of implementation (numbers) proposed for the POC during the first year of development.

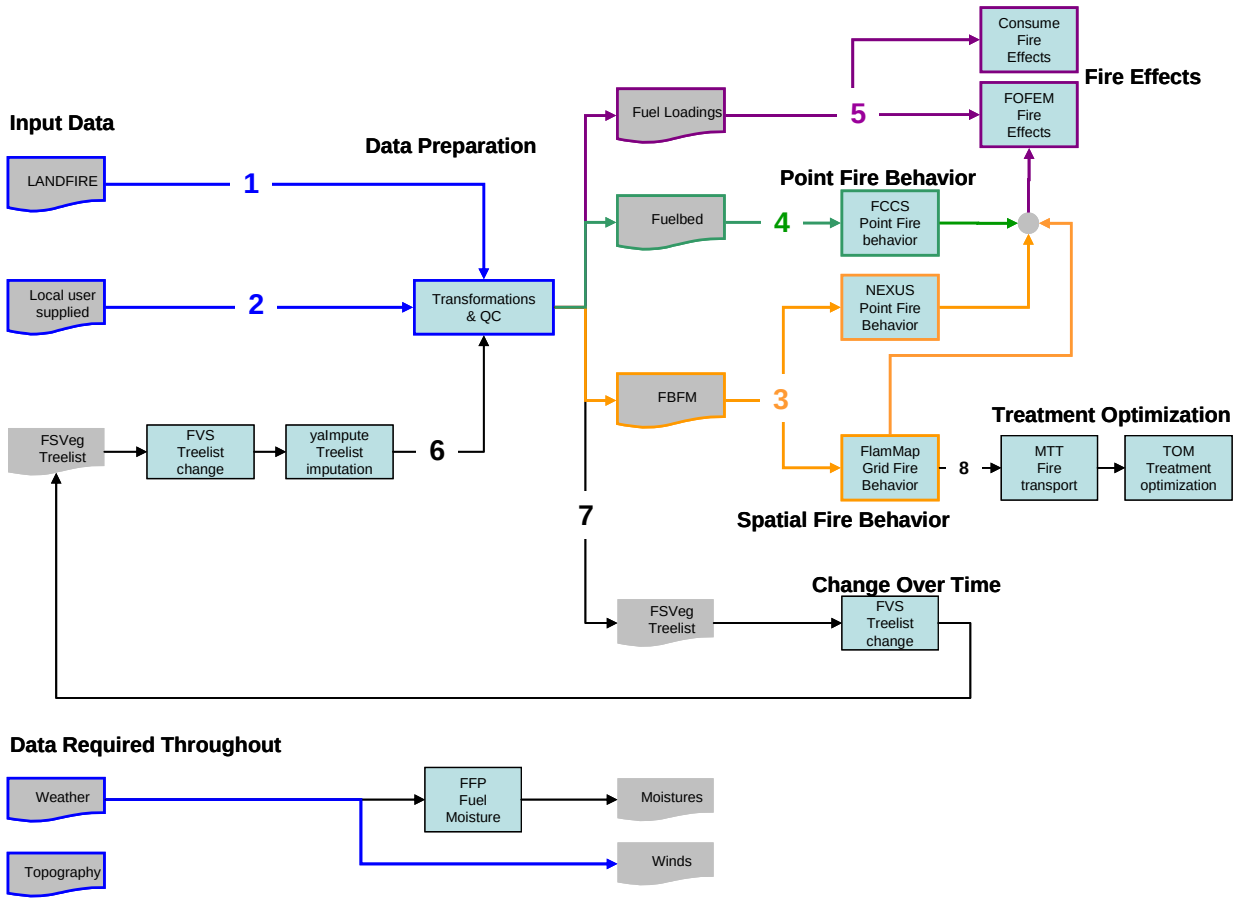


Figure 2-8. Illustration of the pathways (colored lines) and implementation order (numbers) for the IFT-DSS POC.[7]

The main objective of the IFT-DSS POC development effort is to deliver a system that is immediately useful for the fuels treatment planning community. To satisfy this objective, we chose to implement the work flow scenarios that have been identified, on the basis of feedback from the fuels treatment planning community, as the most immediately relevant.

[7] The pathways indicated by black numbers are proposed for the second year of development. It should be noted that the risk assessment work flow scenario is not shown on this diagram; however, the risk assessment portion of the system will be implemented in the second year of development.

2.8 IFT-DSS POC MODEL INTEGRATION

The various data and software applications used in the fire and fuels community differ in manner of invocation, robustness, generality, types of modeling, execution platform, and in other ways. Integrating these applications in the early stages of IFT-DSS development poses both technical and structural challenges because the applications are not in standard formats for integration. However, an important community development goal is, over time, to adopt simple software development protocols and guidelines that facilitate the integration of scientific applications into the various SOA systems in the fire and fuels domain (e.g., IFT-DSS, BlueSky Framework, and WFDSS).

Developers of individual applications are likely to have limited time, if any, to integrate their products with the proposed framework. Therefore, tools and guidance will be developed to facilitate collaboration, including:

- software tools to simplify and streamline the process of integrating new models into the IFT-DSS;
- technical assistance to application developers, including software application programming interface (API) documentation and email or phone support;
- easy registration of components and simplified delivery of applications and updates to users; and
- clear specifications for data standards, and specifications of required APIs that the software applications are expected to support.

In some cases, the original developers of key applications that are necessary for integration with the IFT-DSS may be unavailable or unable to make software modifications. In these cases, assuming the source code is available in the public domain and can be decoded, the IFT-DSS development team will integrate the application by:

- obtaining the source code, making necessary modifications, and integrating the application into the IFT-DSS;
- “wrapping” the original software application in a wrapper application that is itself integrated into the framework; or
- re-engineering the application in a way to allow it to integrate directly into the framework.

In the early stages of the IFT-DSS POC development effort, the software applications will require integration into the IFT-DSS by one of the methods described above. Over time, when a sufficient number of key applications support the new framework, it is likely that there will be significant user pressure on the developers of remaining applications (and new applications being developed) to provide voluntary support for the framework.

For the IFT-DSS POC, three existing fire and fuels models will be integrated: FlamMap, NEXUS, and FCCS. When these applications are integrated with the IFT-DSS, their functionality will be available within the system; however, they will be accessed by the user

through the IFT-DSS GUI rather than the GUI of each individual application. The following explanations show how these applications will be integrated into the IFT-DSS POC:

- **FlamMap** will be integrated by obtaining source code and dynamic link library (.dll) files and making modifications necessary for integration with the IFT-DSS. The IFT-DSS development team communicated with the developers of FlamMap, who agreed to provide FlamMap .dll files and support for making modifications to the FlamMap source code so that it can be integrated into the IFT-DSS POC.
- **NEXUS** will be integrated using the "wrapping" approach described above.
- The **FCCS** will be re-engineered for easy integration with the IFT-DSS. It is fortuitous that the FCCS is currently being redesigned to make the desktop application more extendable, flexible, and maintainable. As a result, the newly designed FCCS will be easily integrated with the IFT-DSS POC.

Fortunately, the integration of the three applications listed above will use all three integration approaches (described in more detail in Section 4). This diversity of approaches will provide the opportunity to develop protocols, guidelines, and possibly an integration toolkit for the science development community. The protocols, guidelines, and toolkit developed during the first year of POC development can be tested early in the second year of IFT-DSS development as new applications are integrated.

In the IFT-DSS POC, applications added to the system will become available within the system and can be viewed in a resource library. The resource library will contain information about the application, its intended use, and authorship. During the first year of development, applications will be added by the IFT-DSS development team. By the end of the second year of development, a science collaboration engine that allows science developers to register applications within the system, assuming that their applications adhere to the integration protocols and guidelines, will be implemented.

2.9 IFT-DSS CONNECTIONS TO OTHER SYSTEMS

The IFT-DSS POC will also explore the initial connections to two other SOA systems in the fire and fuels community: the BlueSky Framework and WFDSS. The BlueSky Framework has recently been expanded to provide access to its processes through web services. IFT-DSS will leverage BlueSky services where possible. For example, the BlueSky Framework provides ready access to the Consume fire effects model. IFT-DSS may implement the Consume model by calling out to the Consume service provided by the BlueSky Framework. The IFT-DSS development team will also work closely with the WFDSS development team to share technology. There is overlap in functionality between the two systems, and IFT-DSS will reuse the work done by the WFDSS team to the maximum practical extent. Identified areas of overlap include the data viewing and editing tools, access to LANDFIRE data, and data preparation for fire behavior models.

3. THE IFT-DSS POC USER EXPERIENCE

An important aspect of the IFT-DSS architecture is system behavior and user experience. This section describes how the IFT-DSS POC will behave and how the user will interact with the IFT-DSS POC. Draft mock-up screen shots are included to show what the GUI screens will contain.

The IFT-DSS POC GUI will be web-based and will function within a standard web browser (e.g., Internet Explorer, Firefox, etc.). A user login screen on the IFT-DSS POC website will be the main point of entry into the system. The first time that users log into the IFT-DSS POC, they will be prompted to create a user profile that will capture each user's email address, password, job title, organization, and other relevant information. This user profile will be saved and used for subsequent logins.

From a user perspective, a project analysis will involve three phases: (1) project setup and planning; (2) software model execution and iterative analysis; and (3) project finalization, documentation, and archive. The remainder of this section describes the general experience of an IFT-DSS POC user during each of the three phases of a project analysis. It should be noted that the GUI mock-ups are in draft form and will change as feedback is received.

3.1 LOGGING INTO THE IFT-DSS

The IFT-DSS POC user interface will provide a portal for a user to create project analyses, execute modeling scenarios, view the results of modeling scenarios, and manage and share projects with other IFT-DSS users. The user interface will be secured and accessed by using an email address and password. **Figure 3-1** shows a mock-up of the IFT-DSS POC login screen.

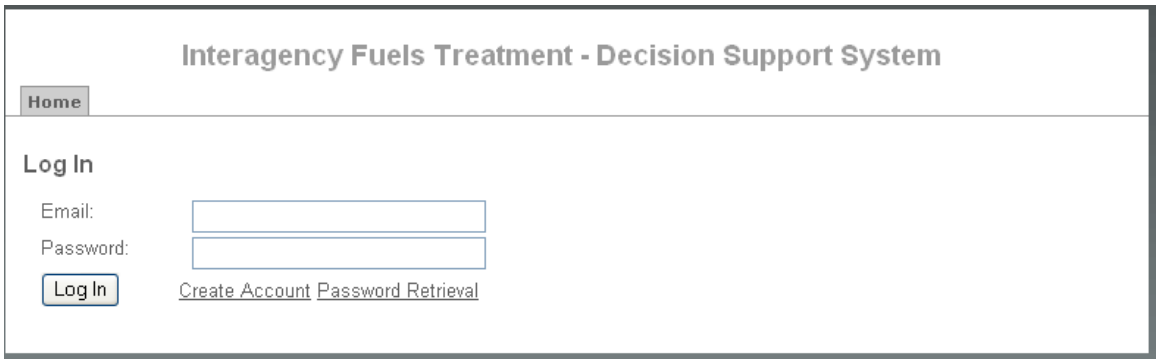


Figure 3-1. Mock-up of the IFT-DSS user login screen.

Once a user logs into the system, he or she will be able to create an account, a user profile, and a personalized home page. This home page will be the user's central location from which he or she can create a new project analysis, revisit past analyses, manage analysis projects,

and view or edit the user profile. **Figure 3-2** shows a mock-up of the IFT-DSS POC user home page. **Figure 3-3** shows a mock-up of the "My Profile" screen where users manage their profile information.

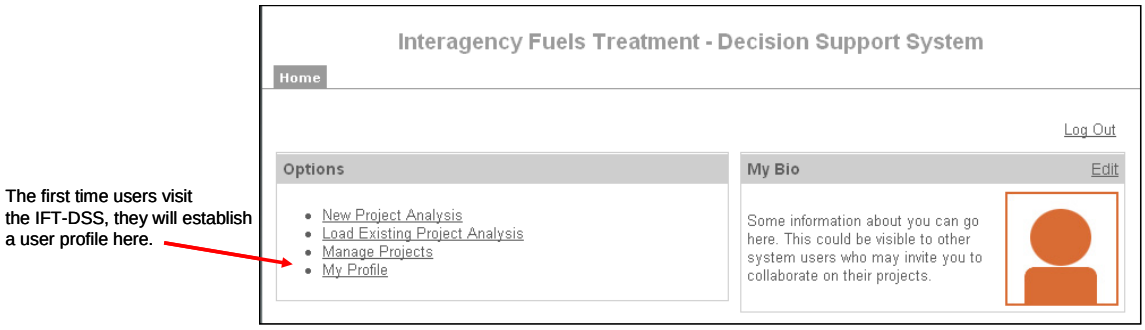


Figure 3-2. Screen shot of the IFT-DSS POC user home page.

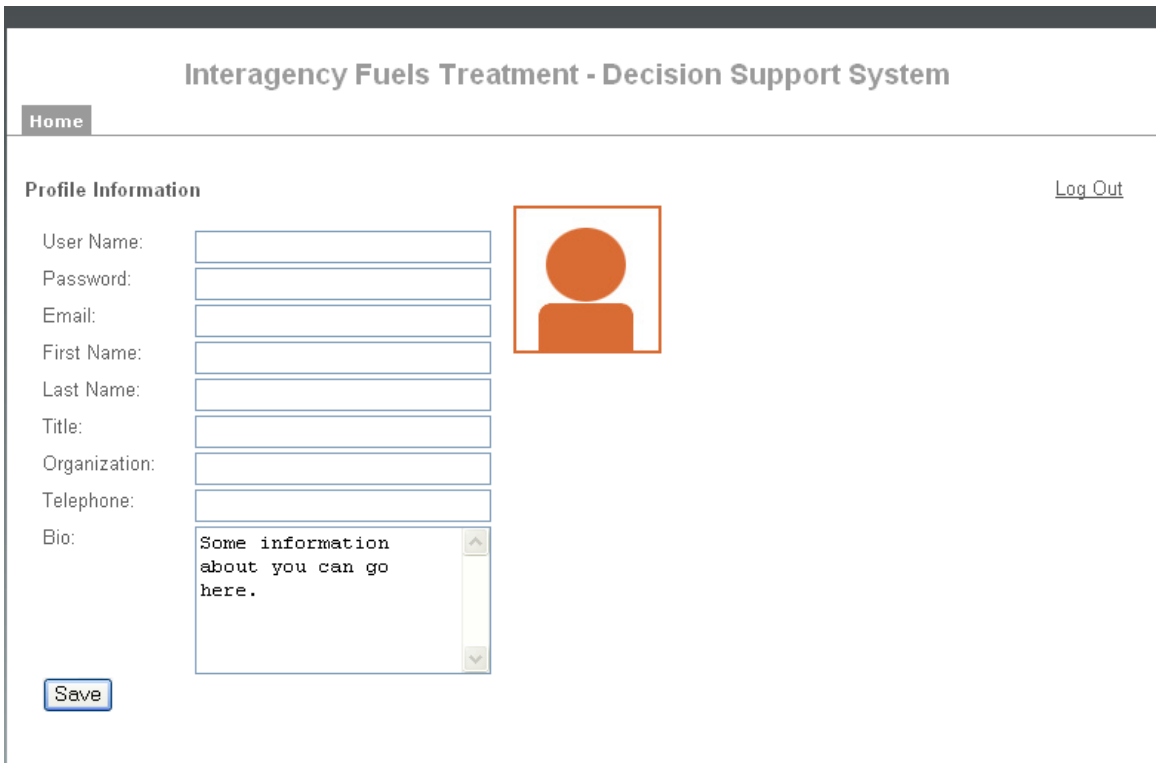


Figure 3-3. Screen shot of the "My Profile" screen where users manage their profile information.

User profile information will be stored in a table in the Scenario Database (discussed in Section 4), and will be used by the system for user authentication. All project analyses created by a user will be associated with a user record in the Scenario Database. The user will have the

ability to log out of the system at any point during an analysis session without losing information from that session. When a user logs out of the system, he or she will be re-directed to the login screen.

3.2 PHASE I – PROJECT SETUP AND PLANNING

After a user logs into the system and reaches his or her home page, he or she will be provided a list of options. From this list, the user can create a new analysis project, load an existing analysis project or modeling scenario, manage projects, and edit or view his or her user profile. **Figure 3-4** shows a mock-up of the user options available on a user's home page.

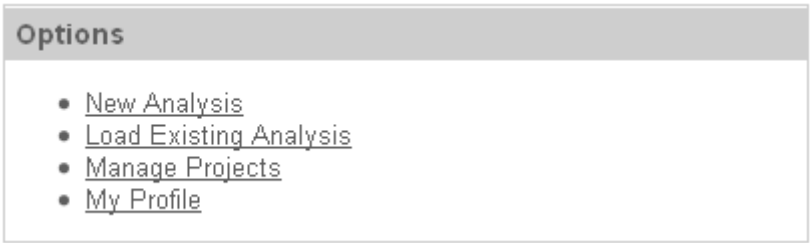


Figure 3-4. Mock-up of the user options available on a user's home page.

Upon selecting the "New Analysis" link, the user will be presented with a project setup screen that will guide the user through the creation of a new project analysis. **Figure 3-5** shows a mock-up of the project setup screen.

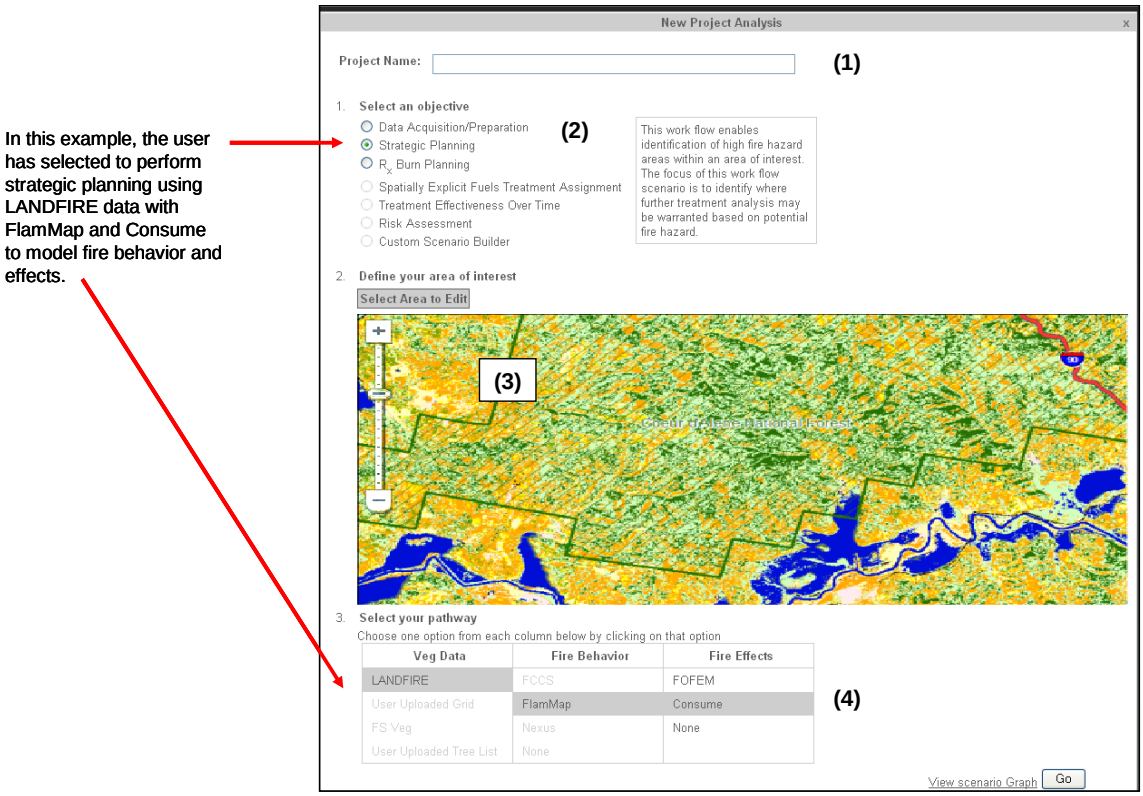


Figure 3-5. Mock-up of the IFT-DSS POC project setup screen.

A user creating a new project analysis would go through four general steps:

1. providing a project name;
2. selecting an analysis objective;
3. defining an area of interest; and
4. selecting a modeling scenario pathway.

In the first step, the user will provide a name for the project analysis that will identify the project throughout the system. Next, the user will select an objective and be presented with the option to enter a description of the project analysis and any other relevant information associated with the project. After selecting an objective, the user will select an area of interest (AOI), using a map viewer tool that will allow the user to draw a box on the map that generally defines the AOI.

For the fourth step, the system will provide model scenario pathways appropriate to the selected objective. The user will select one of these pathways with a point-and-click select tool.

3.3 PHASE II – SOFTWARE MODEL EXECUTION AND ITERATIVE ANALYSIS

Once a user has set up an analysis, the navigation elements of the website will change on the basis of the selected modeling scenario, and the specific pathway for the selected scenario will be displayed as a tabbed navigation bar. A dynamically generated scenario graph based on the selected pathway will also appear. The scenario graph describes the data inputs, outputs, and processes involved in running the selected modeling scenario. **Figure 3-6** shows a mock-up of the dynamically generated tabbed navigation bar and scenario graph for a sample modeling scenario.

The modeling scenario graph will be interactive, allowing the user to click on a step in the graph and navigate to a data input screen associated with that step. For example, if a user clicks on the "FlamMap" oval in Figure 3-6, a screen prompting the user for the information needed to run the FlamMap model will appear. The scenario graph will also provide feedback to the user by graphically depicting how far the user has progressed through the steps of the modeling scenario run at any given time. The purpose of the modeling scenario graph is threefold: (1) to visually describe the inputs, outputs, and processes associated with a selected modeling scenario pathway, (2) to allow the user to access the input screens for the selected modeling scenario pathway, and (3) to report the status of a modeling scenario run once the run has been executed.

The pathway that defines the modeling scenario will be stored in the Scenario Database and dynamically retrieved by a program called "pageCreator" at run time. The pageCreator's role will be to interact with the database and retrieve pages and elements for each page, based on the selected modeling scenario. The pageCreator will also dynamically generate a scenario graph (Figure 3-6) for the modeling scenario selected.

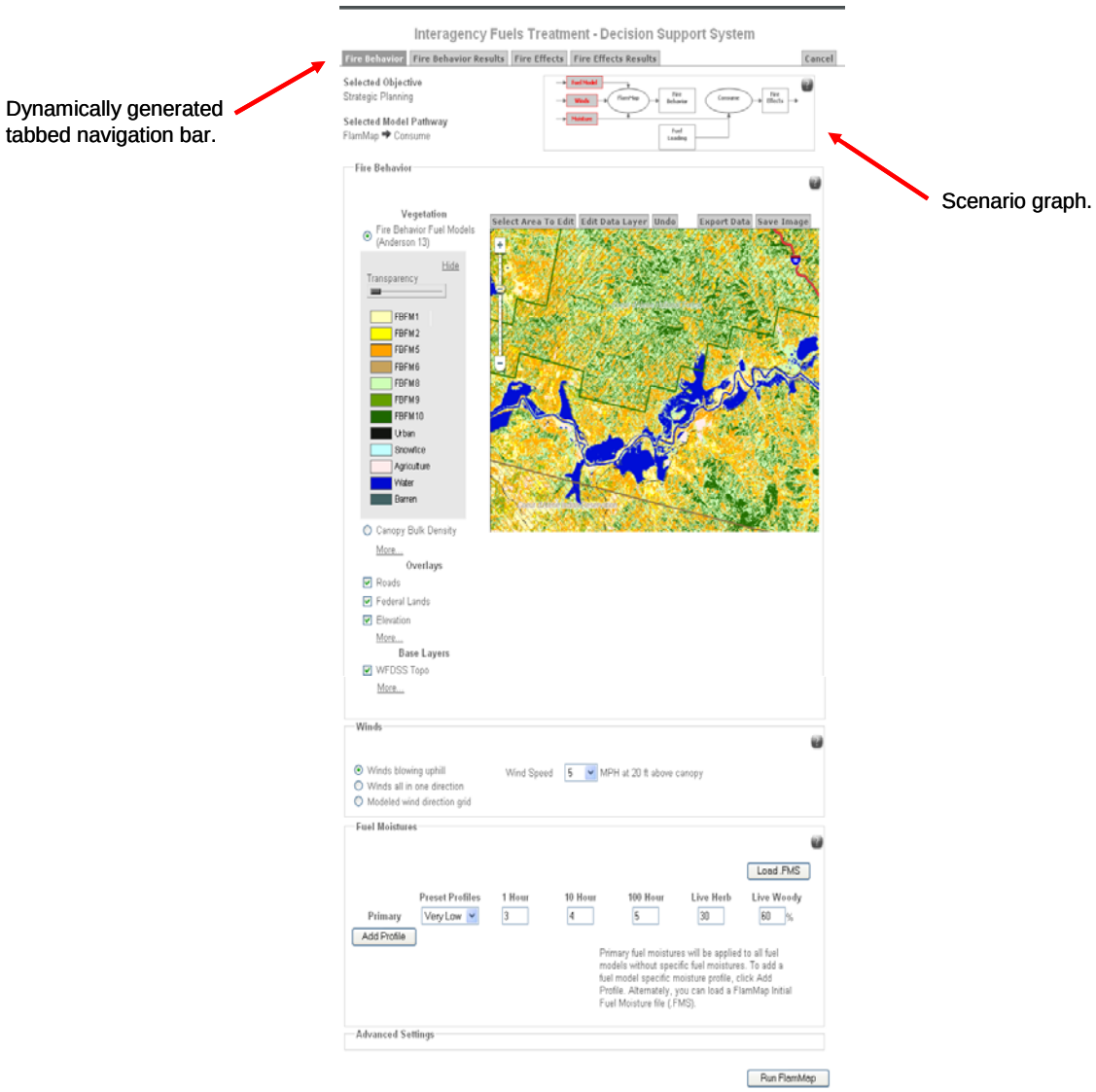


Figure 3-6. Mock-up of the dynamically generated tabbed navigation bar and scenario graph for an example modeling scenario.

At this point, the user can navigate through the modeling scenario pathway by clicking through the tabbed navigation tool (as shown in Figure 3-6) or by using the dynamically generated modeling scenario graph (Figure 3-6). These styles of navigation allow the user to iterate the steps of the modeling scenario. At each step, the user will be able to provide input or accept default values by using a range of input tools such as text field inputs or a GIS-based visualization and editing tool

At each step in the modeling scenario pathway, the user will also be able to execute an individual model by clicking on a "run" button on the model input screen (lower right corner of Figure 3-6). The outputs from the model will be displayed on the following screen (see the

example of fireline intensity from FlamMap in Figure 3-7) and will become the inputs to the next model or process in the modeling scenario pathway.

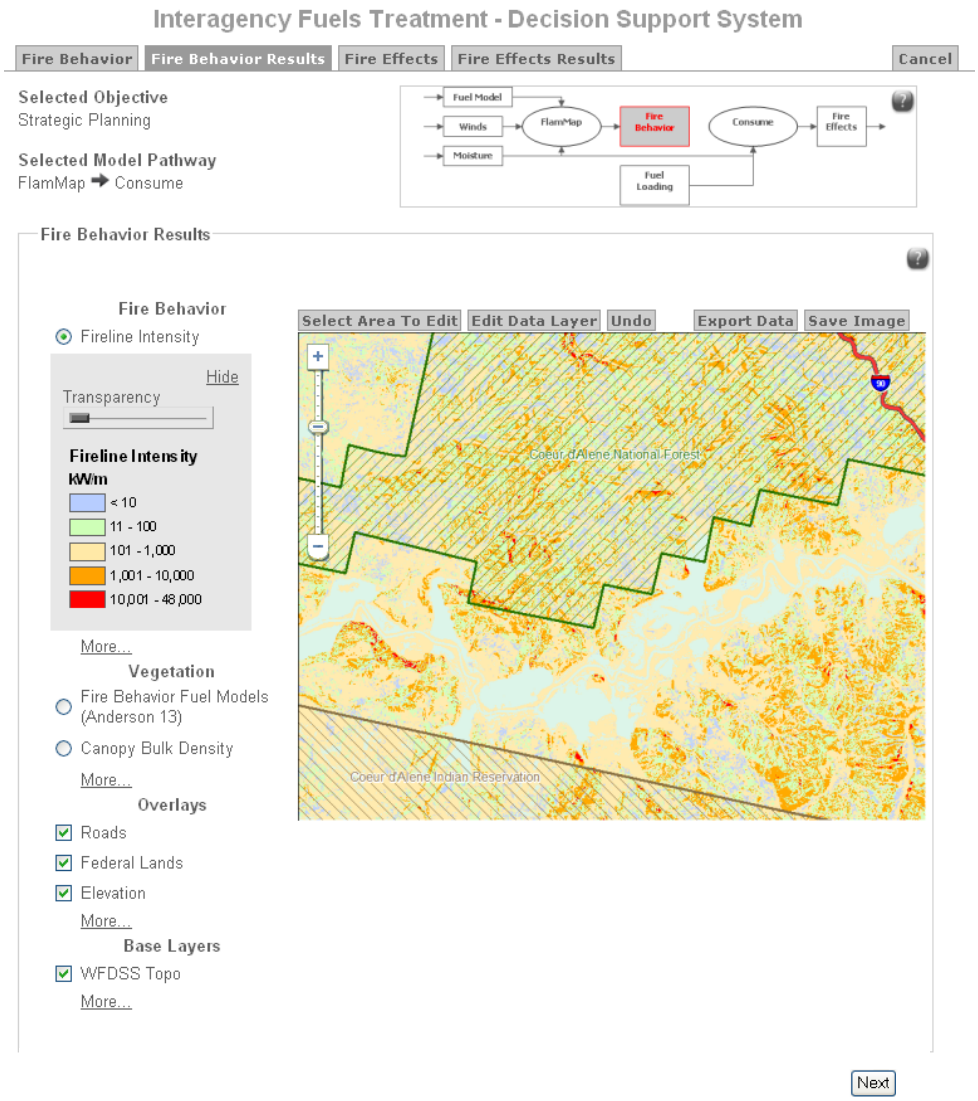


Figure 3-7. Example results of fireline intensity from FlamMap.

As a project analysis is performed, the IFT-DSS will track the status of model run scenario processing time. The user will be able to view status updates on their user home page or receive email status notifications for processes that require a long period of time to complete. The email notifications will include a link connecting the user to the completed process step, allowing the user to continue stepping through the scenario from where they left off.

Figures 3-8 and **3-9** show the screens that a user would navigate to perform prescribed burn planning analysis using FlamMap to model fire behavior and Consume to model fire effects.

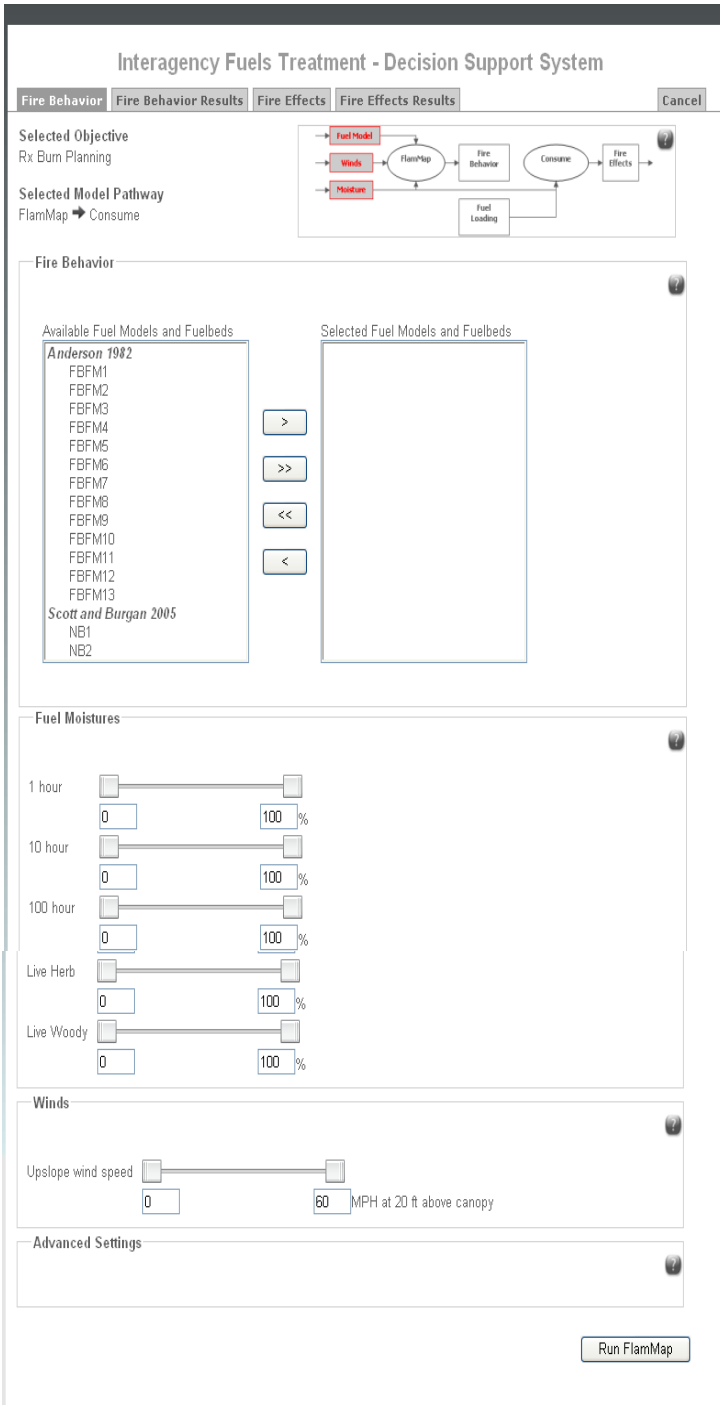


Figure 3-8. Example page showing input options to model fire behavior for a range of weather conditions using FlamMap.

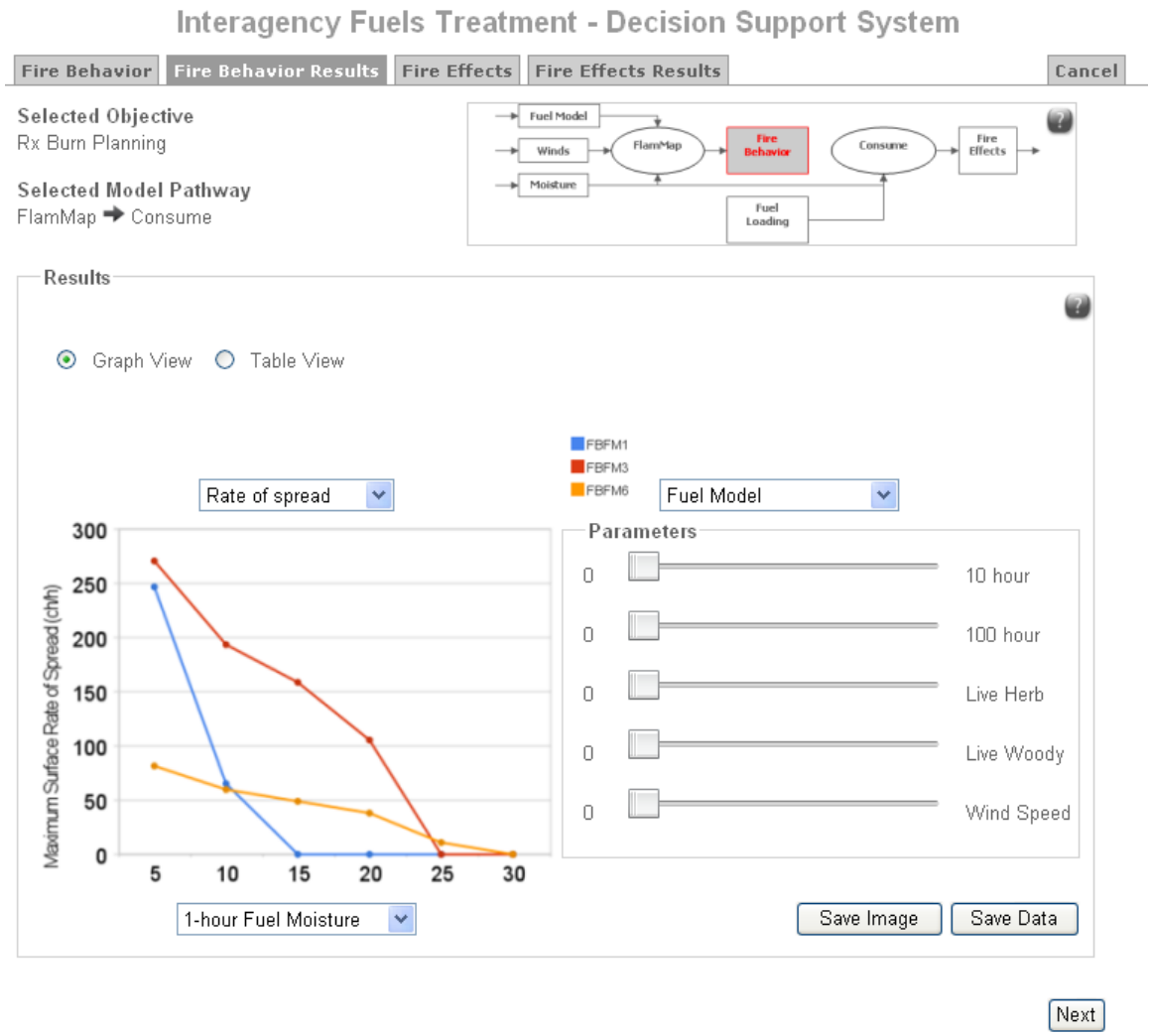


Figure 3-9. Example page for exploring fire effects modeling results for a range of weather conditions using Consume.

3.4 PHASE III – PROJECT FINALIZATION, DOCUMENTATION, AND ARCHIVE

After reaching the end of a modeling scenario, the user will be able to save the scenario as a "run" in the project analysis. The project will then be accessible from the "Manage Projects" screen shown in **Figure 3-10**. The Manage Projects screen allows the user to view projects, view runs within a project, view project results, and assign sharing privileges to other users. For example, a user could designate another user as a contributor who would have fewer (or read-only) privileges than the project owner. The user can also archive a project from this screen.

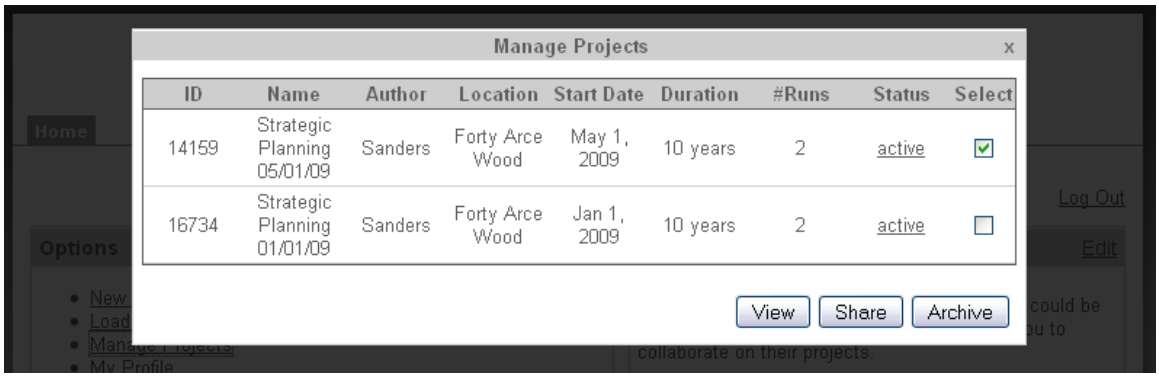


Figure 3-10. Example Manage Projects screen in the IFT-DSS POC with two project analyses displayed.

From the Manage Projects screen (Figure 3-10), the user will be able to view and edit a project or modeling scenario run, share a project with other users and specify their user privileges, archive a project, and generate reports related to a specific project. The archived projects will continue to be available to the user but will no longer be considered active. See Section 4.1.9 for details regarding the report generation for a project.

4. TECHNICAL SOFTWARE DESIGN

This section presents the IFT-DSS POC software design in a manner that is generic and almost completely independent of programming language, hardware, and operating system implementation decisions; that is, the design can be realized in a variety of ways using any one of many hardware and operating system configurations. It is important to realize that this design was constructed in a way that system components can be developed in different languages and use different operating foundations while still conforming to this design. Approaching the design in this manner allows the software architect to maintain a maximum degree of modularity when considering how the system will be constructed. Section 5 contains the language, hardware, and operating system details initially proposed for the IFT-DSS POC.

The presentation of the IFT-DSS POC software design in this section is separated into three parts: (1) a description of the system components, (2) a description of the connections among these components, and (3) a description of the behavior of the system as a whole. **Figure 4-1** is a diagram of the overall structure and components of the system. The structural components are labeled with letters, and the software interfaces are labeled with numbers. Each of the nine components, A through I, is described in Sections 4.1.1 through 4.1.9; each of the seven software interfaces, labeled 1 through 7, are described in Sections 4.2.1 through 4.2.7. Section 4.3 revisits the components and software interfaces and describes how they interact.

Components

A. Models – The scientific and computational components of the system; all other components support the model operations.
B. Model Adaptors – Enable integration of different models into the system
C. Geodatabase (GDB) – Stores the actual data inputted to and outputted from models
D. Data Sequencer – Relays data between the Geodatabase and models
E. Executive – Directs the execution of the models
F. Scenario Database – Stores representations of how different "runs" of models are orchestrated
G. Navigator Web Application – Visualizes spatial data, allows users to edit data values for model inputs, allows users to execute runs through scenarios or parts of scenarios, all from a web browser
H. Project and Planning Database – Stores administrative data about fuels treatment projects
I. Planner Web Application– Enables users to manage fuels treatment projects from a web browser

Interfaces

1. Model Data Exchange – Connections between machines running models to relay model inputs and outputs
2. Model Control – Connections between the Executive and model adaptors to prepare the model data exchange connections
3. Sequencer Control and Monitoring – Interaction between the Data Sequencer and Executive to manage scenario execution

4. Executive Scenario Polling and Feedback – Executive queries of the Scenario Database for required data, and feedback about model and machine execution performance
5. Executive Launching and Progress Reporting – Web application signaling of the Executive to compile and run a scenario segment; and the Executive reporting progress back to the web application
6. Map and Data Presentation Interface – Communication between the Geodatabase and the web application components
7. Web Page Generator and Database Interface – Web application queries of the Scenario Database to produce the web pages displayed to the user for model inputs and outputs

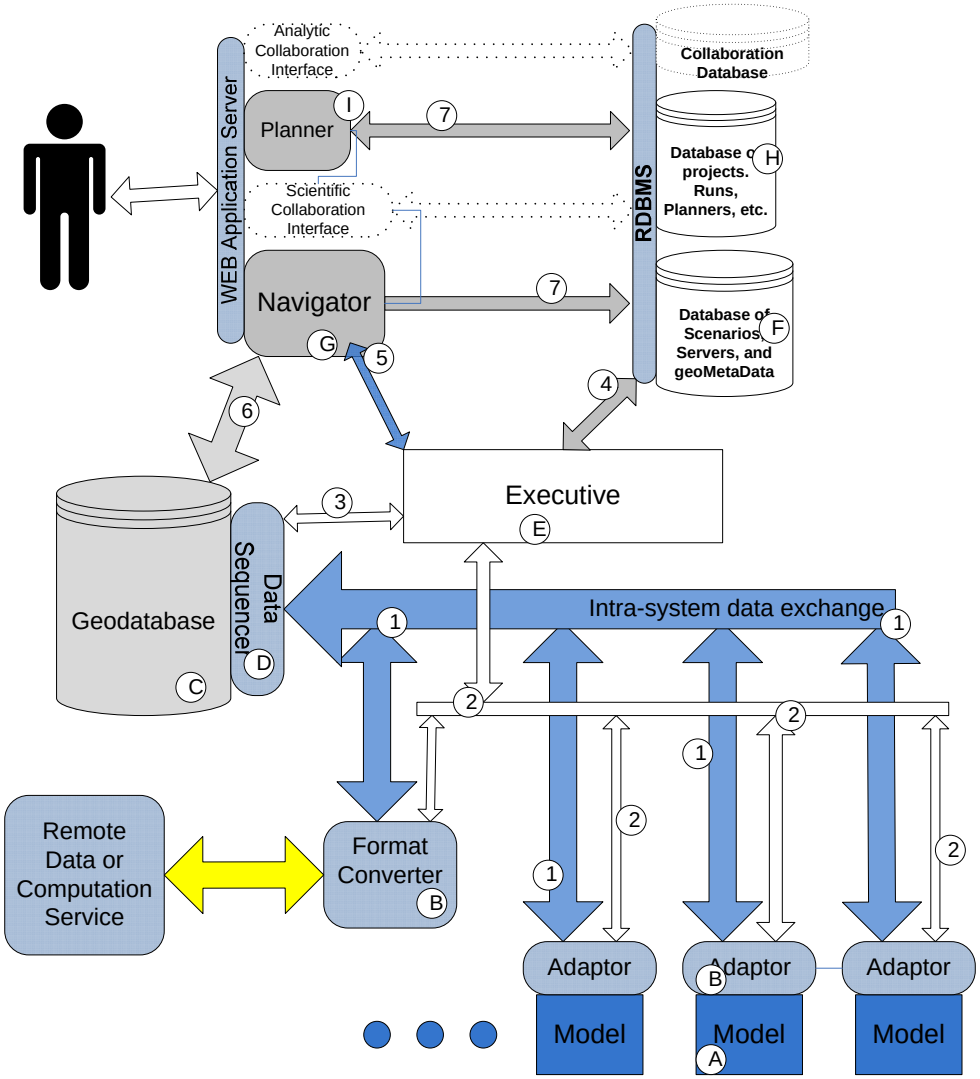


Figure 4-1. Overall structure of the IFT-DSS depicting the system's major components (with lettered indicators) and software interfaces (numbered). Features that will not be implemented in the IFT-DSS POC are represented with dashed borders.

All of the system components will be realized in the IFT-DSS POC with two exceptions. The collaboration interfaces will not be fully realized, as indicated with dashed borders in Figure 4-1. Other components will be fundamentally complete, but some of the more complex scenario execution optimizations described in section 4 may not be present.

Experience gained from using the initial version of the POC will help determine priorities for optimizing existing features and adding new ones. It may prove to be more valuable to add new models and scenarios than to maximize resource utilization through parallelization of model execution. Some of the optimizations will only become possible when system deployment configuration and run execution times are known. The features involved in these optimizations reside primarily within the executive, data sequencer, and model adaptors.

The collaboration interfaces include support for analytic collaboration among planners sharing knowledge and effort and for scientific collaboration in the authoring of models and model scenarios. These two components will not be part of the initial POC system. The analytic collaboration interface will be implemented early in year 2 of development. At that time, support for scientific collaboration will also be added through implementation of the scenario editor and other tools for software development.

The analytical collaboration interface is the mechanism by which planners work together on projects. This includes control of shared access to project data and runs. A project owner will be able to solicit input from selected peers. Planners will also be able to discuss the preliminary results through a project-specific message relay or discussion forum. This will augment the planning project results by encouraging planners to work together within the system. It will also document quality assurance by recording peer feedback with the other project data.

The scientific collaboration interface provides a mechanism for advanced users, data providers, and software developers to add new capabilities to the system. This will include a scenario editor and a software development environment. The scenario editor is the mechanism by which advanced users can construct custom analysis pathways. Users of the initial IFT-DSS POC GUI will be able to select an analysis objective and choose from the predefined processing scenarios supporting that objective. The scenario editor will extend this capability by allowing users to select data and model sequences to create custom analysis pathways.

During the first year, the IFT-DSS POC development team will concurrently integrate the initial set of modeling capabilities and produce the environment in which these models are developed and run. During this period, they will also solicit feedback from the scientific developer community to identify the software development tools that will complete the analytical collaboration interface.

4.1 COMPONENTS

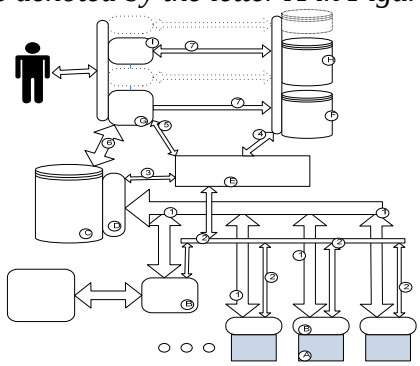
The components described in Sections 4.1.1 through 4.1.9 are software programs and subsystems. It is possible to realize this design with a variety of hardware topologies. Some of

the components, such as the distinct databases and the web application components, may be implemented as a single entity that embodies a combination of them all.

4.1.1 Models

Models constitute the heart of the system, as they are the scientific and computational components of the system. All of the other system components were designed to support the operation of the models and the examination of the model outputs.

***Models** are denoted by the letter **A** in Figure 4-1.*



The first models to be integrated into the IFT-DSS POC will be programs (or parts of programs) that already exist and are in use in the fire and fuels community. These models include FlamMap, NEXUS, FCCS, Consume, and FOFEM. These models will be integrated into the IFT-DSS POC by the methods described in Section 4.1.2.

In the future, as protocols and guidelines for integrating models into the IFT-DSS become available to model developers, it will be much easier for a model to be integrated into the IFT-DSS. Future models will be better aligned with IFT-DSS's execution paradigm, resulting in several advantages: they will be much smaller than any standalone modeling program, they will be simpler to develop and maintain, and they will be able to take full advantage of the system's parallel processing architecture.

Developers will be able to integrate models into the IFT-DSS by using the "Model Subclass" method described in Section 4.1.2. In this simple and efficient approach to incorporating models into the system, future models may consist of small sections of programming code that define and execute specific, self-contained functions. For example, many existing standalone software applications contain programmed functions that are linked together and tightly coupled to a user interface. In the future, it will be more desirable for model developers to program functions as discrete pieces of programming code that are unlinked to other functions and are decoupled from a user interface. Ideally, they will consist of a relatively small number of lines of program code embodying a single behavior or very few modeling equations. This approach will allow new functions and features to be easily integrated into the IFT-DSS without the need to redesign the IFT-DSS software framework or user interface.

A simple example of a model is one that computes slope and aspect given elevation. For example, given a raster grid of elevations with square cells of length S on a side as shown in **Table 4-1**, the function that computes the slope for the central cell [2,2] is shown as pseudocode in **Table 4-2** in the context of extending the model parent class. (The model parent class is predefined, and there is only one model parent class for all models.) This function is the only

code that the developer will have write in order to add the slope calculator "model" to the IFT-DSS system.

Table 4-1. Representation of an array of grid cells containing elevation values.

Z_{11}	Z_{21}	Z_{31}
Z_{12}	Z_{22}	Z_{32}
Z_{13}	Z_{23}	Z_{33}

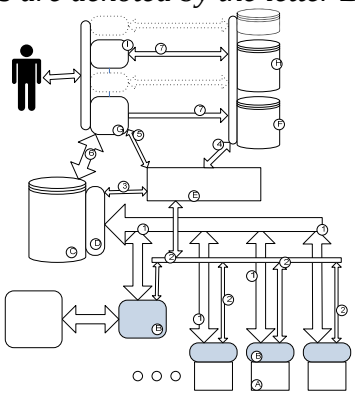
Table 4-2. Pseudocode for an example slope/aspect model.

```
Class ElevationToSlopeModel extends ModelParent
     Override ExecuteModel(neighborElevations)
          // calculate the slope and aspect

          // calculate east-west gradient
          dX = S*((Z31 + 2Z32 + Z33) – (Z11 + 2Z12 + Z13))/8

          // calculate north-south gradient
          dY = S*((Z11 + 2Z21 + Z31) – (Z13 + 2Z23 + Z33))/8

          slope = sqrt(dX^2 + dY^2)
          aspect = tan^-1(dX/dY)
          return an object containing slope and aspect
     end ExecuteModel
end ElevationToSlopeModel
```

4.1.2 Model Adaptors

The model adaptors constitute an environment—a layer of software—that makes it possible for models to be added to the IFT-DSS and function as services. The model environment will consist of several sublayers and will encapsulate two software interface protocol controls. **Figure 4-2** below illustrates how these layers and protocols enable three methods of adding models to the IFT-DSS. Each method is a different type of model adaptor.

Model Adapters *are denoted by the letter* ***B*** *in Figure 4-1.*



One protocol control (depicted by the large arrow above the Data Sockets box) is used to exchange data among models within the system and to interface with the Geodatabase (C in Figure 4-1). The other protocol control, depicted in Figure 4-2 by the small arrows above the OTS (off-the-shelf) Application Server boxes, is used to initiate the model's processing and guide the configuration of data sockets. These two protocol controls correspond to software interfaces 1 and 2 in Figure 4-1 and are described in Section 4.2.1 and 4.2.2. All of the models in the IFT-DSS will use the same two software interface protocol controls, and these software interfaces will be exposed to the rest of the IFT-DSS system.

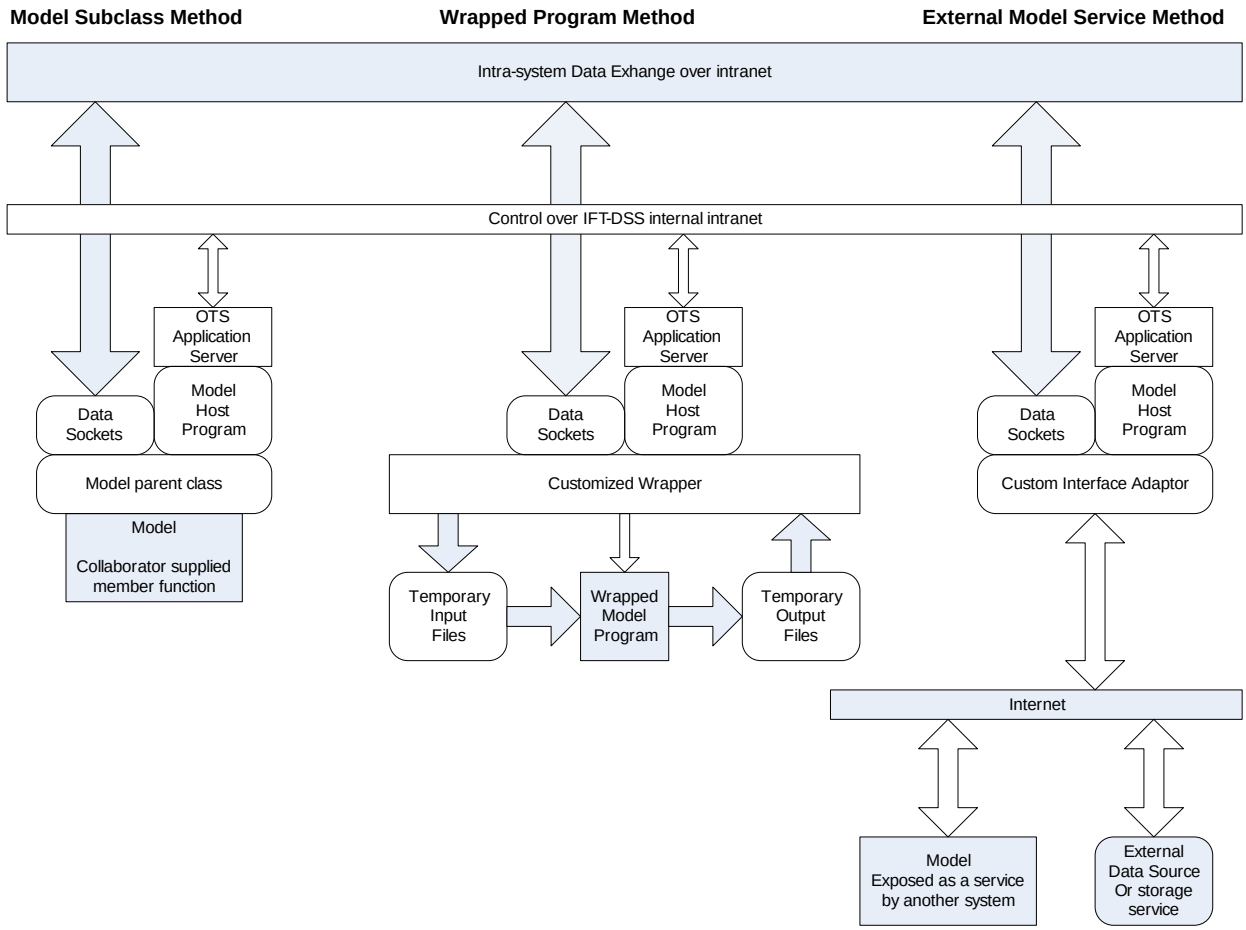


Figure 4-2. Illustration of the three model hosting methods by which models can be integrated into the IFT-DSS and function as services.

Model Integration Methods

As shown in Figure 4-2, there will be three types of model adaptors, i.e., three ways by which a model can be added to the IFT-DSS.

1. Model Subclass Method – The adaptor type on the left depicts the most efficient method. It consists of a standard API that includes an application server, a hosting program, communication handlers, and all of the data handling code required to operate the modeling code. The model is added to this code base by extending (creating a subclass of) the model parent class. The other "wrapping" methods employ these same components (except the model parent class).
2. Wrapped Program Method – The middle adaptor type depicts a more expedient method for deploying existing model programs by providing a wrapper program that communicates with an existing model program and translates the model inputs and outputs to the format standards used by the IFT-DSS.

3. External Model Service Method – The adaptor type depicted on the right side of Figure 4-2. This method allows the IFT-DSS system to incorporate functionality from other modeling systems.

These methods all share a number of important features. Near the top of each module stack depicted in Figure 4-2 are three shared features represented by rectangles labeled "OTS Application Server", "Model Host Program", and "Data Sockets". Below each group are various forms of "Model" components.

Model Control Communications

The first component required for service oriented model hosting is an application server. Our system will employ an "off-the-shelf" solution for this requirement. The purpose of this component is to receive messages from an external control, interpret them as requests for model execution, and act on them by deploying the model host program. "Models" are computational objects, at the heart of each model integration method, that run on the distributed compute servers of the IFT-DSS. The OTS service provider causes the models to run, passes them arguments and commands, and relays status information. Model hosting programs receive the control messages from the Executive (E in Figure 4-1). The protocol of the commands is described in Section 4.2.2. The application server communicates status information from the model components back to the Executive through the same interface.

Model Data Communications

In addition to command and control data, the models will communicate model data on a peer-to-peer basis. These data pass among models running on distributed computational servers, the GDB, and other possible (interface) subsystems. See **Figure 4-3** below for a Unified Modeling Language (UML) diagram corresponding to the following description of model communication infrastructure.

Each model program will have zero or more input streams and zero or more output streams. Each input stream will de-multiplex data from one or more connections to an output stream of another model. Likewise, each output stream will multiplex or broadcast through one or more connections to the input streams of other models. Information about the number of streams, their data type, their fan-out or fan-in, and the specific addresses of each channel socket will be received from the Executive.

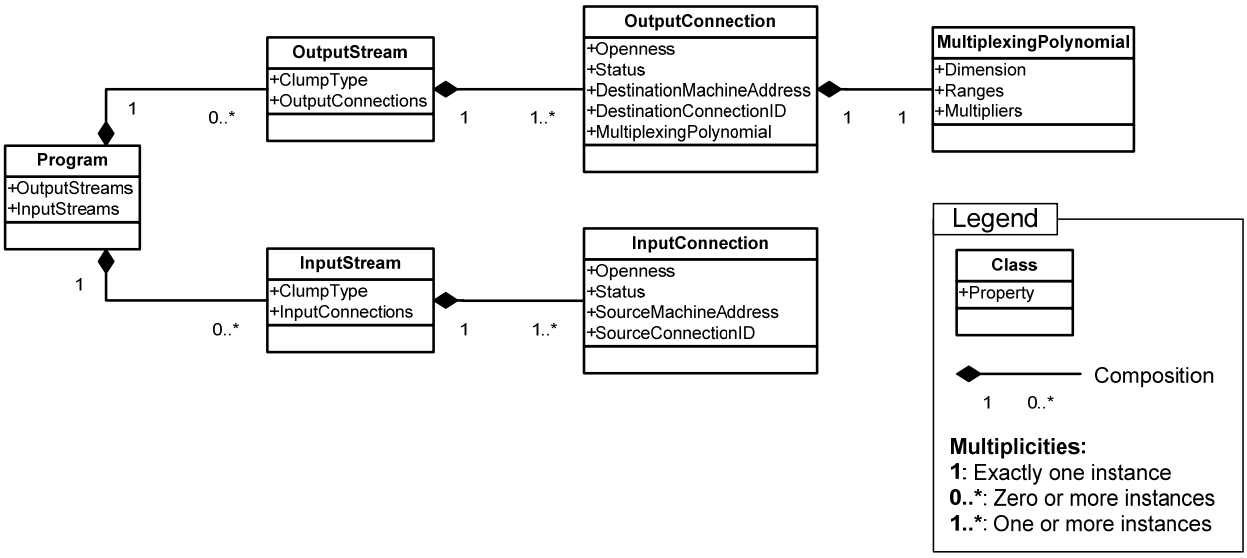


Figure 4-3. UML diagram of the model communication infrastructure.

Each output connection has a "multiplexing polynomial," which allows parts of a model to be run on different machines. It is also possible to do the same work on multiple machines or to use the output from one step as the input to several later steps. Likewise, data can simultaneously be sent to the next modeling step and to the Geodatabase for archiving. This data distribution capability is achieved by the member functions of the input and output stream classes, which use the multiplexing polynomial to sequence data from an output stream to a set of input streams. These classes have members common to stream-like objects including open(), close(), read(), and write(). However, the actual behaviors of these class members are different from those of their familiar analogues. To illustrate these differences, **Table 4-3** shows the pseudocode for the output stream members.

Table 4-3. Pseudocode for key member functions to the output stream class.

```
Open()
      For each connection
            Save multiplexing polynomial coefficients and ranges
            Fork and in the new thread
                  Note the time
                  Open the connection
            End thread
      End for
End of open()
```

```
Close()
      For each open connection
            Terminate the connection (i.e. session)
      End for
End of close()
```

```
writeObject()
      format the object into a packet
      for each destination connection
            check object index against multiplex polynomial
            if index is in a write window
                  fork
                        wait until connection is ready
                                                   send the packet
through the connection
                        note the time
                  end thread
            end if
      end for
end of writeObject()
```

The open() and writeObject() member functions must use separate threads for each connection because each of these can require some time to synchronize with the other end of the connection. In the pseudocode this is represented with “fork” and “end thread” clauses. None of the actual programming languages that may be used to implement these behaviors actually has an analogous construct.

Input streams will have open() and close() behaviors similar to the corresponding behaviors of output streams. But, there will be no need for a channel select polynomial. All of the data that arrive at an input channel will be processed and the input streams will have a readObject() behavior instead of a writeObject() behavior. ReadObject() will read one object from any of the one or more open connections. **Table 4-4** shows the pseudo code for the input stream member.

Table 4-4. Pseudocode of the readObject member function of the input stream class.

```
      readObject()
            for each input socket
                  check for pending data
                  note how long it's been waiting
            end for
            read the packet that has been waiting the longest
            extract the data object from the packet
            if an object with this packets index has already been
received
                  discard packet
                  start over from the top
            else
                  return the received object
            end if
      end of readObject()
```

The stream classes handle the peer-to-peer flow of science data. In addition, the model host program must communicate control and status information with the Executive and cause the model to be performed for each set of data inputs. This part of this program is the same for all models.

The processing here is independent of output multiplexing, but it must take into account an analogous property of inputs. Each input stream must have a known "front aperture" description. The front aperture is a specification of the neighborhood of locations at which parameters must be supplied to the model in order for the model to calculate its result. The front aperture of many (aspatial) models is a delta function. In other cases, it may span the entire domain of the input. Both of these conditions can be indicated with flags. In other cases, ranges of values will have to be specified for each input coordinate.

Processing entails running the model many times. The sequencing of model execution is determined by the parameters of the program. One form of sequencing will be to execute the model once for each record in a specified ("indexing") input stream. Indexing streams may have to be constrained to have only delta function entrance pupils. In some cases, there may be no suitable input stream. When this occurs, a special form of input stream is required. This pseudo-input stream would have no connections but would respond to readObject() calls with indexed null data rows.

The model host program is responsible for configuring peer-to-peer connections with other models, managing all communications, and executing the model as needed. A very brief pseudocode of this component is presented in **Table 4-5**. The data handling for each model is divided into phases. Only a single phase is required, but generally, three I/O phases will correspond to prerequisite, concurrent, and retrospective communications. More phases are possible, and there is no benefit in constraining the range of supported I/O phases. Zero or one model functions will be executed during each model phase.

During each processing phase, all of the outputs are initiated, and a separate processing thread is produced to handle each input. The input threads each initiate the communications associated with a single input stream, receive and process each data object, and finally terminate the communications. The processing done by these threads is limited to unpacking received data objects and distributing their content into the appropriate areas of large, queue-like data structure where model input objects are staged. Whenever the data insertion results in the completion of a staged data object, a message is sent to the main thread. The main thread, upon receiving notification from one of its daughter threads, causes the model's routine to execute and process the staged input data. Any output objects that result are transmitted. Monitoring of staging queue object completeness continues until all of the input threads terminate. Then the main loop finishes and either moves on to the next phase or terminates.

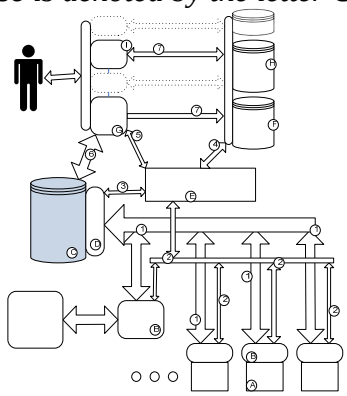
Table 4-5. Pseudocode of the model hosting program.

```
Model Hosting Program:
   Parse all control parameters from dispatcher
   For each phase of the model
             Open each output steam
             For each input stream
         Fork and in the new thread
                   Open
                   WHILE (still objects to read)
               Read in an object
                     Unpack
               For each effected ingest queue object (including new
ones)
                   Distribute data into ingest queue object
                   If ingest queue object is 'complete' (i.e.
aperture full)
                     Send message parent thread
                   End if
               End for effected queue entries
             END WHILE
             Close
               End of thread
       End for all input stream
       WHILE (more 'aperture full" messages expected)
         Wait for 'aperture full' message from daughter thread
         Do the model operation(s)
         For each output stream
             Write a result object
         End for
             END WHILE
             Close everything
   End for phase
END of model hosting program
```

4.1.3 Geodatabase

The Geodatabase (GDB) is used to store a special form of geographic information system (GIS) data. In addition to the spatial properties common to a GIS, the GDB will manage data with non-spatial domains. It will manage raster images from the LANDFIRE database, tree-list and polygon stand layers such as those used by the Forest Service's Forest Vegetation Simulator (FVS) software, and other forms of multidimensional data. The GDB will also handle time series and parameter ensemble data sets.

*The **Geodatabase** is denoted by the letter **C** in Figure 4-1.*



The IFT-DSS POC will have relatively modest GIS requirements compared to those that must be added as the system matures. For the IFT-DSS POC, the GDB will be required only to manage raster spatial data; however, the Geodatabase in the POC system will be designed to handle additional data types that will be needed as the system matures. The GDB will house these data (images) and any ancillary data required to manage them. All of the data for a single study will be stored at the same spatial resolution and in the same map projection. The images will contain bands for all of the parameters provided by LANDFIRE plus those produced by FlamMap, FOFEM, and Consume.

FCCS parameters (standard and a limited number of custom fuelbed parameters) will only be supported through lookup table indirection. Indirection is the ability to reference something using a name, reference, or container instead of the value itself. FCCS image files will contain only pixel values that correspond to indices in that table and not the actual fuelbed data themselves. When a fuelbed is modified, either the modification will be shared by every location in the study that is assigned that fuelbed, or a new fuelbed will be created. The user interface will, therefore, have to support two modes of modifying spatial data. It must be possible to modify or create fuelbeds and to reassign what fuelbed a specific location is associated with. A similar, indirect, storage structure may be used for other raster data types (including LANDFIRE-like layers) as well. Later, when support for polygonal data sets is added, a similar, indirect method will be employed.

The data structures used to manage layers where the above indirect coding is not used will be remarkably similar to those used to house the indirect values. A table of parameters will exist in either case. This table will house unique records in the indirect mode and a much larger number of non-unique ones in the direct mode.

In addition to supporting the map layers for modeling, the GDB will accept ancillary layers from the users and present these with the other map layers. These layers will assist the user in navigation and could be included in reports.

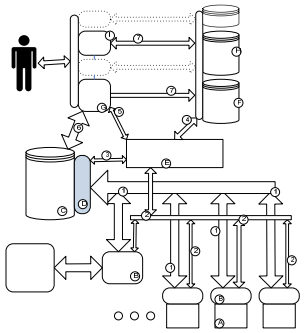
Every processed map layer will also have a hidden layer containing the time of last update. These timestamp layers, and the production of processing maps based on them, will be among the most important features of the GDB. The generation of masks (that indicate where the content of one layer is newer than the content of another) is an important early step in the compilation of scenario fragments. If this operation is not readily supported by the GDB, similar functionality will have to be built into the Data Sequencer described in the following section. Additional binary morphologic operations will likewise have to be performed. Most notable among these will be dilation of masks to reflect the projection of the mask through a model's front aperture.

Upon initiation of a run, the GDB will be directed to generate working data sets from pre-existing standard data, user supplied layers, or default value sets. The system will have access to a complete copy of the LANDFIRE data. These map layers will serve as the default spatial data layers unless the user has provided study-specific images containing the same parameters. As the system's capabilities expand through the addition of new models and scenarios, the library of default imagery will have to expand to support the additional model inputs.

4.1.4 Data Sequencer

The Data Sequencer is an interface between the models and the geographic information system where the modeling inputs and outputs are stored. Since the sequencer determines what data are processed, it is responsible for managing key aspects of system load and is thus an important component of the system.

*The **Data Sequencer** is denoted by the letter **D** in Figure 4-1.*



Beyond simple speed considerations, this component has the opportunity to maximize performance by limiting the waste and redundancy in the data processed by the models. If performance constraints merit, the component will exploit these additional opportunities. Although the initial version of this program is not likely to have these features, the design does account for them, and their implementation will be prioritized along with expansion of the initial suite of modeling capabilities.

It is desirable to ensure that only required data are processed. When only a small section of the input data needs to be processed (as is the case when models must be re-executed following edits), the models' front aperture description will be used to ensure that only the required areas of the input data are reprocessed. This check requires cooperation between the Data Sequencer, which has access to relative data ages, and the Executive, which has the apertures and controls the topology of the compute service network.

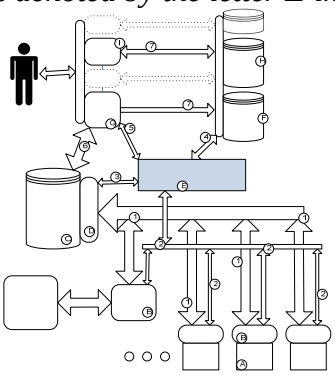
A later version of this component may further improve processing efficiency by determining which data within the edited regions are redundant and therefore do not need to be processed. This type of optimization is most easily incorporated into the model hosting programs, but the gains may be greater if implemented in the Data Sequencer component.

Prior to process initiation, a series of messages will pass between the Executive and the Data Sequencer. These will include communication about masks that depict the areas affected by data updates. By working with the Executive (described in the next section) and the GDB to process these data, this program will generate input and output masks for the affected areas of the scenario fragment. The Executive (after distributing the scenario fragment across the pool of computational resources) will send information to the Data Sequencer that the sequencer will use to make the necessary input and output connections and plan the distribution of data to the appropriate outputs. Finally, the Data Sequencer will receive information from the Executive upon initiation of the execution of a scenario fragment. This information will include the identity of the run, a list of the run fragments saved results, and a list of required inputs and intermediate data sets. The inputs and results lists will trigger the generation of write and read sockets as described earlier. The interaction described above is summarized in Section 4.3.3 as a swim lane diagram (**Figure 4-11)**.

4.1.5 Executive

The Executive component compiles a scenario segment and is invoked in response to an action by the user. A scenario is represented as a Directed Acyclic Graph (DAG). In the DAG, the models are the nodes and the data passed between models are the edges, or vectors. The Executive receives parameters defining which DAG vector of which scenario should be produced.

*The **Executive** is denoted by the letter **E** in Figure 4-1.*



The Executive responds by querying the Scenario Database for additional data about the run's scenario. After analyzing these data, it causes model services to begin running with appropriate peer-to-peer data connections. Then, it directs the GDB Data Sequencer to send the required input data. This, in turn, causes the models to execute and result data to be returned to the GDB. This process is summarized in the brief pseudocode shown in **Table 4-6**.

Table 4-6. Brief pseudocode for the Executive program.

```
Executive program
   Receive scenario (section) from User Interface
   Get model, server, scope, etc data from scenario DB
   Compile a Network of connections and processes
   For each process
         Connect to server and start process
         Relay connections and other configuration data
   End for
   Monitor status and diagnostic feedback
   Notify user (by popup or email)
   Update resource capabilities in Scenario Database
End of program
```

The most complicated part of this program's work is the compilation of a scenario fragment into a set of program execution processes and connections among these processes. This compilation begins with several recursive, binary, spatial operations. As noted above, some of this work will be done by the Executive, and some of it must be delegated to the GeoData Sequencer or the GDB.

Since recursion is difficult in the relational database management system (RDBMS), the scenario's entire set of models and connection will be retrieved from the database. Then the Executive will recursively search the scenario's directed acyclic graph to generate a (possibly redundant) tree-like graph of model nodes and communication vectors. Using code on the

returning side of the recursive calls, the Executive will exhaustively traverse this graph, calling on the GDB or Data Sequencer to generate an input newer than the output mask for each input to each model. If necessary, this step can be optimized or omitted altogether in cases where enough data are modified in the high-order branches to warrant recomputation over the entire domain of the run. Each of these masks is projected forward through the model apertures and onto the output space of the scenario fragment's final results. This combined, dilated mask identifies the part of the results layer that must be recalculated. A final recursion across the DAG in which the output mask is back-projected (with another binary dilation) from each model output to the corresponding inputs is performed. This final mask-building process will use model latency to take advantage of any temporary, intermediate result layers that may have been saved.

Next, any redundancies in the masks are removed (by "or" operations) and the area of each mask is integrated. These areas, multiplied by difficulty estimates for each model, give the system fairly accurate estimates of the size of the modeling tasks that are to be done.

If no single task is very large and the number of computers available is greater than the number of tasks, or if all of the tasks combined add up to a small amount of work, then each model is assigned to the computer most able to do it in order of model difficulty.

Conversely, if it is worthwhile to use a larger network of computers to execute some parts of the scenario fragment in parallel, the work is divided into a large number of approximately equal-sized small tasks. These tasks can be assigned to the computers best suited for them based on task difficulty and bandwidth limitations among computers.

The pseudocode in **Table 4-7** shows the compilation steps. The compiler in the initial version of the Executive will not be as complex as the compiler described here. Aperture calculation with a rudimentary implementation will be adequate because all of the models to be incorporated in the IFT-DSS POC will have delta function or space-spanning apertures.

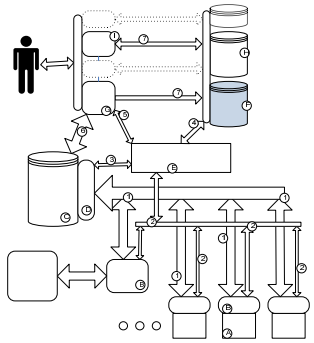
Table 4-7. Pseudocode for compiler component of the Executive.

```
MakeEffortTreeNode(target)
      For each parent of target
            MakeEffortTreeNode(parent)    // RECUR
            getMaskOf(parent younger than target)
            project parent mask through aperture onto target mask
      end for
end MakeEffortTreeNode

BackProjectMaskOf(Target)
      For each parent of target
            project target mask through aperture onto parent mask
            BackProjectMaskOf(Parent)     // RECUR
            Assign parent work phase
                  based on phases of connection between target and Parent
      end for
end BackProjectMaskOf

COMPILER(run, target)
      Get scenario for run
      Make effort tree node(target)
      BackProjectMaskOf(target)
      For any redundant nodes
            Join mask by OR
      End for
      For each node
            Count mask size
            Estimate effort
      End for
      For each work phase
            Use approximate histogram equalization
                  on the distribution of effort estimates
                  to divide the work into multiple, small tasks
      end for
      Partition work space and assign data connections among tasks
      For each task, in inverse order of complexity
            Assign task to the most competent machine
            Remove machine from work phase's resource pool
      End for
END of COMPILER
```

4.1.6 Scenario Database

The Scenario Database stores relational database representations of scenarios and their accompanying DAGs. The Scenario Database provides data to control GUI screen formatting and model execution and is vital to the behavior of the system. The conceptual data model and its influence on system behavior will be discussed in this section.

*The **Scenario Database** is denoted by the letter **F** in Figure 4-1.*



The data model's purpose is to represent a DAG for each scenario (see the conceptual Entity Relationship Diagram (ERD) in **Figure 4-4**). By representing a scenario as a DAG, the Executive can resolve how and when to execute models. The DAG will also be represented in the GUI to facilitate user navigation through a scenario.

Models are fundamental building blocks of a scenario and are represented as nodes in a DAG. A Model entity has inputs and outputs, which are linked together. These links are represented as the data model Vector entity. To ensure that models' inputs and outputs are validly linked, each input and output maps to a ParameterGroup entity. A ParameterGroup is a collection of related Parameter entity data. An Input ParameterGroup entity must match its Output ParameterGroup entity to provide a valid link or vector.

Model inputs and outputs are fully defined before their use in a scenario by authors who will register their models into the IFT-DSS system. Subsequently, authors can then link these models into scenarios. The ability of users to add models and scenarios via a GUI is outside the scope of the IFT-DSS POC system but will be included in later versions. The IFT-DSS POC will have no method or GUI interface for users to view or edit the Scenario Database. Scenarios in the POC will be created, loaded, and managed by the database administrator.

The Navigator tool will query this database and use the results to control the interactive processing of scenario runs and editing of run data. The Executive will query this database for scenario data and compile this data to control processing. Once completed, the Executive will store performance information in the database.

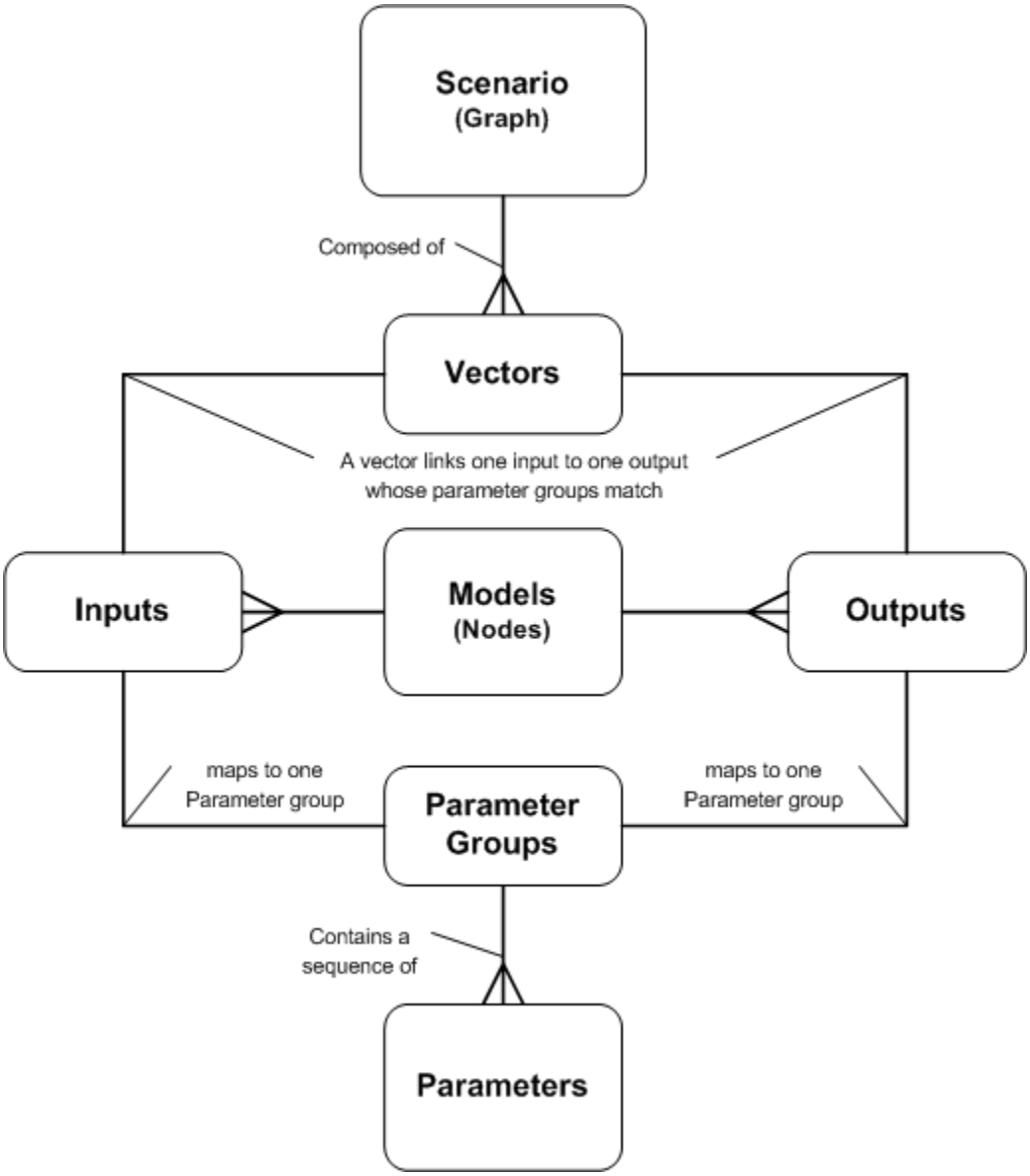


Figure 4-4. Conceptual ERD for the Scenario Database.

The Navigator will query this database and use the results to control the interactive processing of runs and editing of data. The Executive will query this database for ALL of its data related to the specified scenario. The Executive will compile a segment of these data and cause processing to be done. Once completed, the Executive will update the use counters and performance tracking data.

Many behaviors related to this data set, e.g., the ability for users to add models and create scenarios, will be added to later versions of IFT-DSS.

Machines

In order for the Executive to decide how the execution of a model should be distributed across available machines to maximize performance, the Scenario Database has tables (see the conceptual ERD in **Figure 4-5)** describing how well each machine can execute each model (Performance entity). The bandwidth capacity between machines (Bandwidth entity) is also used in this calculation.

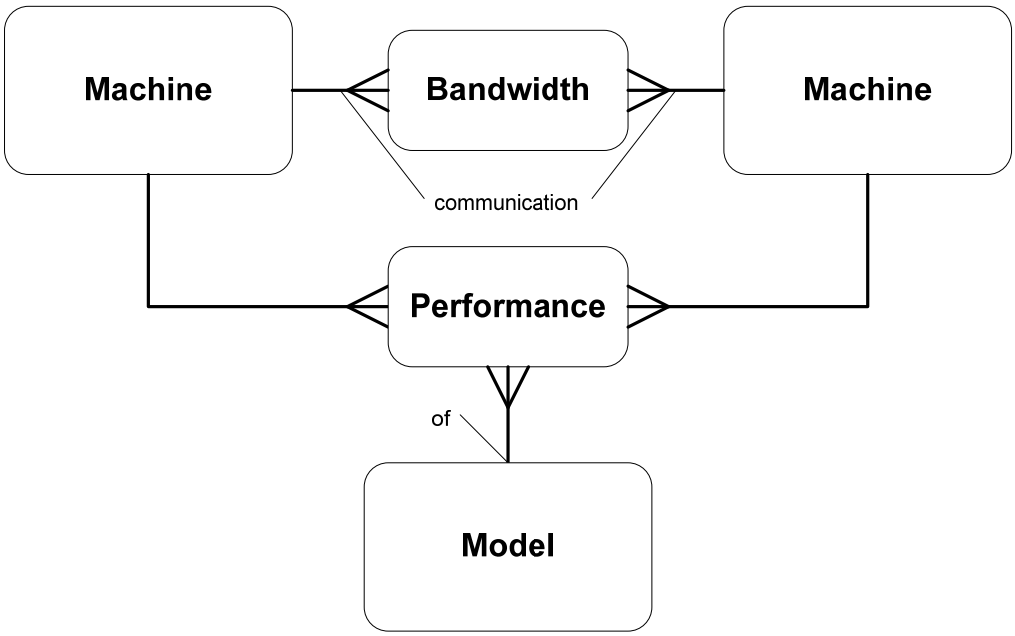


Figure 4-5. Conceptual ERD for subset of the Scenario Database to facilitate efficient model execution.

Templates

To maintain control over the Navigator's GUI format, the Scenario Database includes entities to map the vectors, parameter groups, and their parameters in a scenario DAG to UI templates (see **Figure 4-6**). These templates are HTML documents with placeholders for parameter inputs. This arrangement makes it possible to have customized GUIs for certain scenarios. For example, given a model that takes in winds, moistures, and weather parameter groups, we may have a scenario where all three of those parameter groups appear to the user on a single screen.

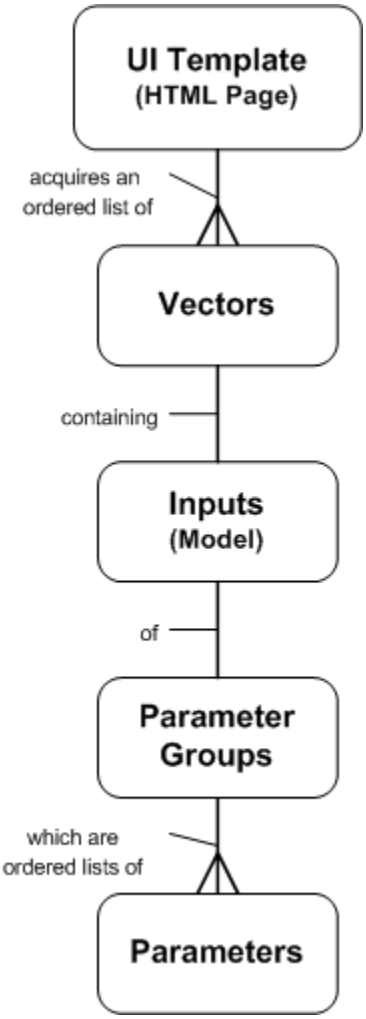
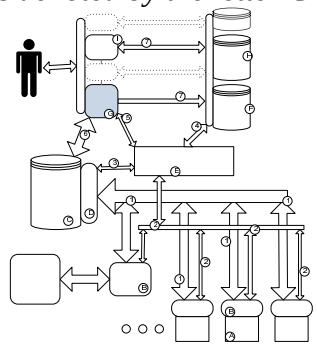


Figure 4-6. Conceptual ERD of user interface HTML templates.

4.1.7 Navigator

The Navigator is a web application that will have two key functions: (1) it will enable users to visualize and explore their data in the form of an interactive map, and (2) it will allow users to edit data values. The tool will function as a data input mechanism (as discussed in Section 3) for user-generated modeling scenario runs. Users will be presented with features that allow them to zoom in and out, pan in different directions, switch between layers and overlays, print maps, and save map images.

*The **Navigator** is denoted by the letter **G** in Figure 4-1.*



In addition, users will have the option to click on a point in the map or draw a polygon to view attribute data for that point or polygon area. Users will also have the ability to edit attribute data associated with the GIS map layers.

As shown in **Figure 4-7,** the interactive Navigator tool will consist of four key subcomponents: (1) a set of underlying GIS map layers, (2) a set of software system instructions that tell the Navigator how to construct the map requested, (3) a map server software component that builds the map, and (4) a subcomponent to add interactive features.

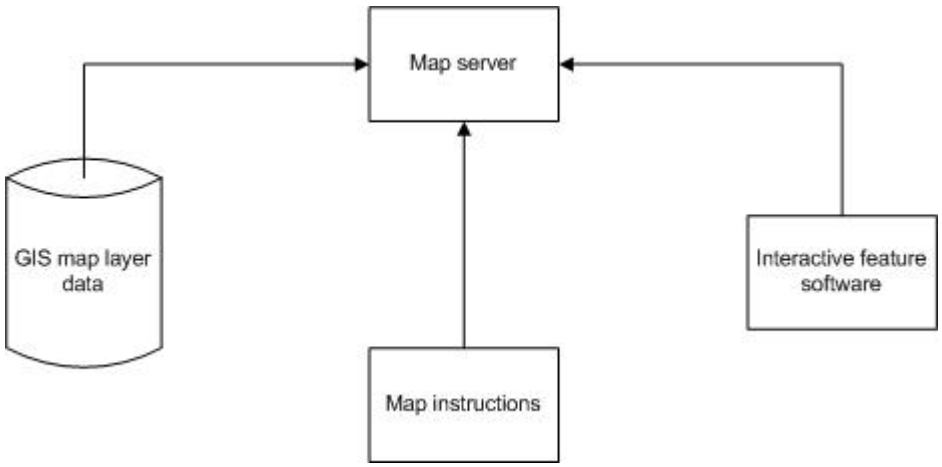


Figure 4-7. Subcomponents of the Navigator.

The data source provides the underlying GIS map layers (i.e., political boundaries, relief, land marks, etc.) for the base map. The map server software will support several data input formats for vector and raster data. For raster data, Tiff/GeoTif and EPPL7 are supported by

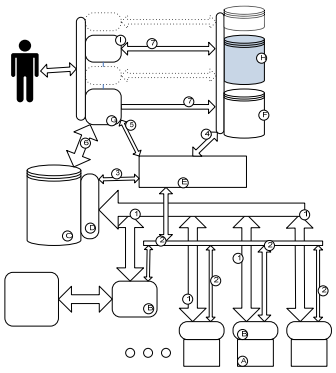
default, but other formats such as GRASS, Jpeg2000, and ArcInfo Grids are supported with the help of the Geospatial Data Abstraction Library (GDAL). In the case of vector data, ESRI shapefiles are the default; however, the software will be compiled to support and read from the Geodatabase, GML files, delimited text files, and more with the vector data access portion of the GDAL library (called OGR).

A set of instructions, in the form of a mapfile, will tell the map server software component where the data source is located and will define how the map should be drawn and displayed. The mapfile is where the map layers and the style of the map are specified. The map server will be a software component that constructs a map image given a data source and a set of instructions. It can be implemented as a Web Map Server, generating maps in response to Web Map Service (WMS) requests. The interactive features of the map will be provided by a software subcomponent. This subcomponent allows maps to be displayed in a web browser with no server-side dependencies. It also implements industry-standard methods for geographic data access, such as the OpenGIS Consortium's WMS and Web Feature Service (WFS) protocols.

4.1.8 Project and Planning Database

The Project and Planning Database will store administrative data about fuels treatment projects. The primary entities are the User and the Project. Users can author models, scenarios, and projects. Users establish permissions for authored projects to control access by other users to their projects and the reports they have generated. Users can act as planners and use projects they have access rights to. A conceptual ERD modeling these relationships is shown in **Figure 4-8.**

*The **Project and Planning Database** is denoted by the letter **H** in Figure 4-1.*



This database will support the Planner web application described in Section 4.1.9. Projects can be accessed by one or more planners. Projects are created by one or more authors, as are the models and scenarios that make up a project. Project authorships are used for managing three kinds of permissions: Project Owner (can do anything), Contributor (fewer privileges than an owner), and Read-Only (can view a project's details but not make changes to it).

A project will also need to reference data in the spatial data system. For example, a project has its area of interest (e.g., Yellowstone National Park). The properties of this area (e.g., elevation, slope, etc.) are stored entirely in the spatial data system, but a project will have to reference this area uniquely so that its properties can be retrieved from the spatial data system. See Section 3 for a description of the user experience that corresponds to this database.

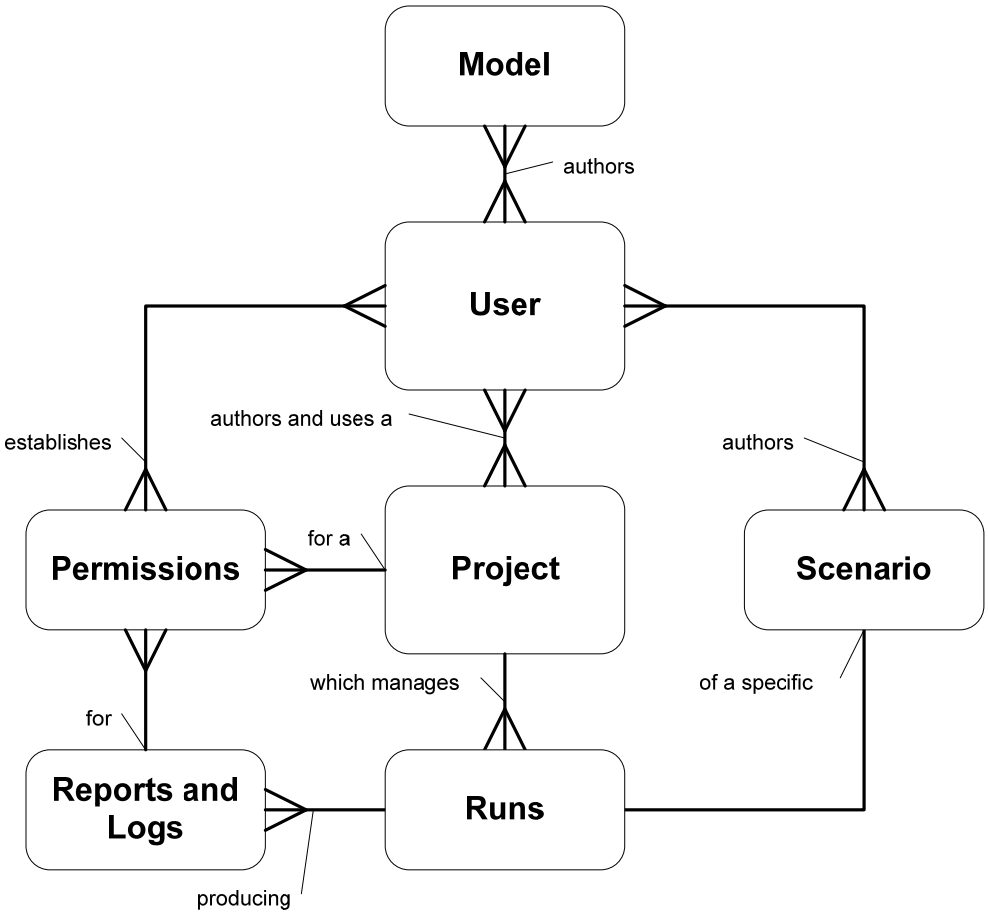
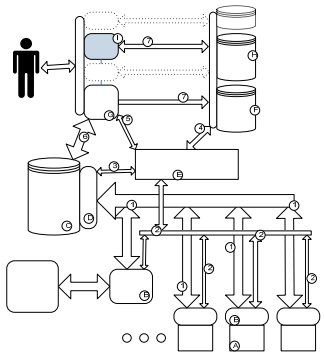


Figure 4-8. Conceptual ERD for Project and Planning Database.

4.1.9 Planner Web Application

The Planner web application manages activities in the IFT-DSS POC, which will include creating a new project or run, loading an existing project or run, sharing a project, and various project management tasks.

*The **Planner Web Application** is denoted by the letter **I** in Figure 4-1.*



Once the user has created a project by defining the run scenario and the area of interest, he or she can launch the Navigator, which presents dynamically generated screens, tabbed navigation, and a scenario graph. The user can then execute runs through the scenario.

After completing the scenario the user can view results, generate reports, and share the

project with other users. A more detailed description of the user experience is presented in Section 3.

Report Generation

When a project reaches a certain point, planners will be ready to compile a report for the project. In the case of a Prescribed Fire Plan report, there exists a Word document template. IFT-DSS POC will have the ability to generate a partially filled-out Word document for this template. Some of the sections of the Prescribed Fire Plan report are blocks of free-form text. The Project and Planning Database will retain this text so that planners can enter the text while logged on to the IFT-DSS and save it for later sessions and ultimately for report generation.

Some of the sections of these reports contain values produced or consumed by certain models (e.g., flame length). Those values, stored in the spatial data system, will be retrieved at report generation time. Users can capture tables, map and graph snapshots for their reports by using the Navigator. For the specific case of a Prescribed Fire Plan report, users will choose to associate such outputs with a report so that the generated Word document will include these snapshots.

4.2 INTERFACES

This section describes the communication among the principal components of the SOA. The interfaces, numbered as in Figure 4-1, are (1) Model Data Exchange, (2) Model Control, (3) Sequencer Control and Monitoring, (4) Executive Scenario Polling and Feedback, (5) Executive Launching and Progress Reporting, (6) Map and Data Presentation Interface, and (7) Web Page Generator and Database Interface. The interfaces can also be listed as follows:

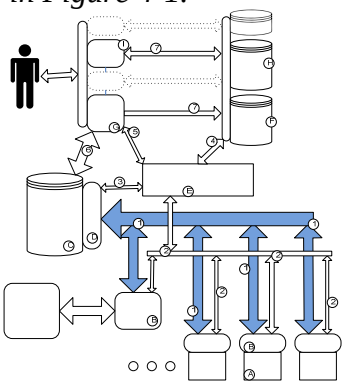
1. Data Sequencer ↔ Model Adaptors
2. Executive ↔ Model Adaptors
3. Executive ↔ Data Sequencer
4. Executive ↔ Scenario Database
5. Executive ↔ Navigator
6. Navigator ↔ Geodatabase
7. Navigator or Planner ↔ Scenario or Project and Planning Databases

The remainder of this section describes the high-level communications protocols for each interface, including a description of each message that passes between components.

4.2.1 Model Data Exchange Interface

The model data exchange interface provides a means of communications between models and with the Data Sequencer. The exchange of data among models is fundamental to the service-oriented model execution and is at the heart of the system. Upon initiation, each model forms peer-to-peer connections with the other models (or the Data Sequencer) that supply or depend on its inputs and outputs. These connections use a simple, efficient protocol.

*The **Model Data Exchange Interface** is denoted by the number **1** in Figure 4-1.*



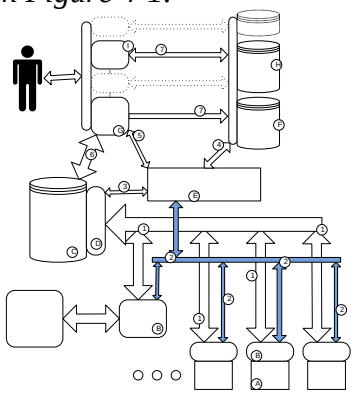
The data are relayed in small packets. Each packet contains a single instance of each member of a "ParameterGroup" from the Scenario Database (see Figure 4-11). Each packet will also contain a locator index unique to the model run, as well as a single data object. All of the objects passed through a given channel will be of the same type.

Output sockets will buffer data for a reasonable period before the remote receiver begins or resumes receiving data. No data will be retransmitted. Packets may be received (and must be processed) in any order. The packet's run locator index uniquely identifies the data spatially and/or temporally. The packet data may contain additional identifying (coordinate) values. It is a logic error for an input channel to receive multiple packets with the same index unless they contain identical data. Race conditions may otherwise result. The effect of these race conditions is not specified.

4.2.2 Model Control Interface

The model control interface allows communications between the Executive and the model adaptors. The Executive and the model adaptors communicate to enable connecting a scenario's models and to execute the scenario, or a portion of the scenario. This protocol is simple in terms of data transmission. It could be implemented as a simple remote procedure call (RPC) or could use HTTP. The content of this exchange is, however, vital to the performance of the system.

*The **Model Control Interface** is denoted by the number **2** in Figure 4-1.*



Once a scenario is compiled by the Executive, the Executive sends startup messages to the models involved in the scenario. Each startup message signals a model to begin listening for incoming data and provides the connection point information required by the model adaptors for hooking up the plumbing between adjacent models in the scenario. A connection between two models is actually between specific input and output streams of the two models, each stream communicating a different data set to or from its model. Moreover, each stream may have multiple connections. When a model completes execution, it notifies the Executive and sends along performance metrics that the Executive will relay to the Scenario Database and use to compile future work. These messages between the Executive and the Model Adapters are shown in **Table 4-8**.

Table 4-8. Messages between the Executive and Model Adaptors.

Messages from Executive to Model Adaptors

RunModel (inputs, outputs) – invokes the model.
Parameters:

```
inputs  – list of input model stream connection info
              per each input:
                  stream ID, IP addresses and port numbers, phase
outputs – output model stream connection info:
              stream ID, IP addresses and port numbers, phase,
              multiplexing polynomial
```

Returns: `status message`

GetProgress – requests progress report
Returns: `progress message`

Messages from Model Adaptors to Executive

NotifyComplete (runstats) – reports runtime performance measurements.
Parameters:

```
runStats – runtime performance metric for this model on this network node
```

4.2.3 Sequencer Control and Monitoring Interface

The sequencer control and monitoring interface allows communications between the Executive and the Data Sequencer. The interactions between the Data Sequencer and the models (included in Section 4.2.1) are complex and critical to the operation of the system. The compiler process and execution of the scenarios are realized by the conversation between the Executive and the Data Sequencer. The protocol that enables this conversation is described in **Table 4-9**, separated into execution messages and compiler messages.

*The **Sequencer Control and Monitoring Interface** is denoted by the number **3** in Figure 4-1.*

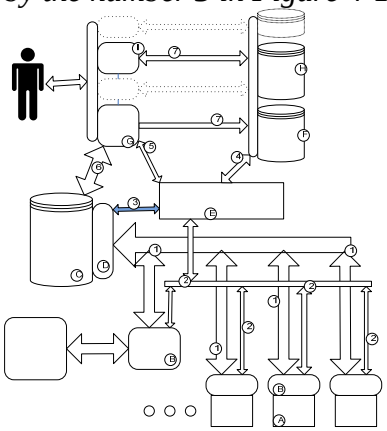


Table 4-9. Messages between the Executive and Data Sequencer.

Messages from Executive to Data Sequencer

Startup – gets Data Sequencer ready to receive messages.
Returns: status message

--- SCENARIO EXECUTION MESSAGES ---

EmitLayerRegion (maskID, layerID, connections, MPs) – tells Data Sequencer to send the part or all of a map layer that is to a set of streams, supplying each stream with its "fan-out"
Parameters:
maskID – the ID of the mask that specifies the region of the layer
layerID – the ID of the layer
connections – list of IP addresses and port numbers to emit to
MPs – list of multiplexing polynomials, one per connection
Returns: status message

ListenForResultLayer (maskID, layerID, connections, MPs) – tells Data Sequencer to send the part or all of a map layer to a set of streams, supplying each stream with its "fan-out"
Parameters:
maskID – the ID of the mask to apply to result
layerID – the ID of the layer in which to store the result
connections – list of IP addresses and port numbers to listen from
Returns: status message

--- COMPILER MESSAGES ---

ApplyAperture (aperture, maskID) – tells Data Sequencer to mark, in the specified mask, the cells that surround the already-marked cells, which are necessary to accommodate the specified aperture.
Parameters:
aperture – the dimensions involved in the map layer and their windows of interest
maskID – the ID of the mask to modify
Returns: status message

MakeMask (input layer, output layer) – tells Data Sequencer to calculate and store in the GDB a mask raster that specifies which cells in some data input layer are older than the same cells in the corresponding data output layer.
Parameters:
input layer – data map input layer
output layer – data map output layer
Returns: the ID of the generated mask

JoinMasks (maskIDs) – creates a new mask that is a combination of the specified masks.
Parameters:
maskIDs – the IDs of the masks to join
Returns: the ID of the generated combination mask.

GetMaskSize (maskID) – queries for number of marked cells in the specified mask.
Parameters:
maskID – the ID of the mask to query
Returns: the number of marked cells

DeleteMask (maskID) – removes the specified mask from the GDB.
Parameters:
maskID – the ID of the mask to delete

Messages from Data Sequencer to Executive

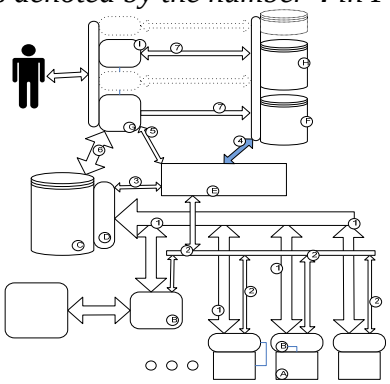
GetProgress – requests progress report

Returns: progress message

4.2.4 Executive Scenario Polling and Feedback Interface

The Executive scenario and polling interface allows communications between the Executive and the Scenario Database. When a user selects the output of a model in the Navigator, the Executive will be told the scenario in question and the model's output that was selected. Subsequently, the Executive will query the Scenario Database (DB) for the data it needs to create a DAG of the scenario (a node list and an edge list).

*The **Executive Scenario Polling and Feedback Interface** is denoted by the number **4** in Figure 4-1.*



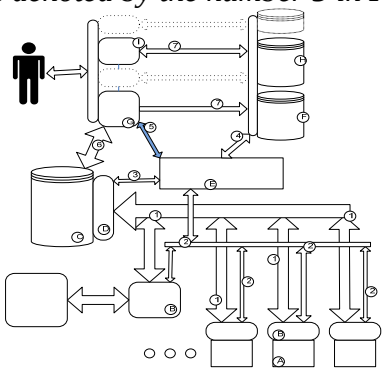
The Executive will also query the scenario DB for information about machines that are available to execute the necessary models for the given scenario. With that information, the Executive will compile an optimal execution plan for the appropriate scenario segment and then initiate execution. After execution of the segment is complete, the Executive reports statistics back to the scenario DB about how well machines executed models in order to improve performance for future runs.

The protocol across this interface will be SQL over the wire (e.g. JDBC). Because the Executive will have to use recursion, it is more appropriate to get a simple and complete node and edge list of the graph from the DB (using SQL, a mostly declarative language) and have the Executive do the rest of the work in a procedural, imperative language (e.g., C++ or Java).

4.2.5 Executive Launching and Progress Reporting Interface

The Executive launching and progress reporting interface allows communications between the Executive and the Navigator. Some user actions in the Navigator will cause the Executive to run and receive progress feedback indicated by the arrow numbered (5) in Figure 4 1. This is triggered by a message from the Navigator that causes the Executive to execute a scenario segment. This process consists of the Navigator sending an RPC to the appropriate machine to launch the Executive.

*The **Executive Launching and Progress Reporting Interface** is denoted by the number **5** in Figure 4-1.*



This RPC will also pass the scenario ID and the point in the overall process where processing is needed. Next, the Executive will compile the proper scenario segment. Once

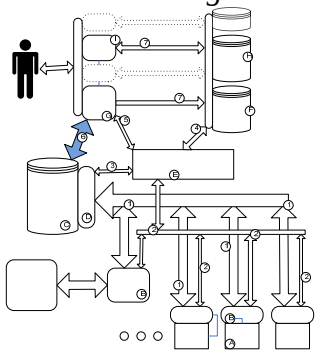
execution begins, the Executive will periodically report progress updates to the Navigator for GUI presentation. In the case of a long execution, the Executive will notify the Navigator so the user knows he or she may log off and receive notification (e.g., via email) when execution has reached completion.

The mechanism for communication between the web application server and the user's web browser may be the Persistent Communications Pattern for Asynchronous JavaScript and XML (AJAX).

4.2.6 Map and Data Presentation Interface

The map data presentation interface allows communications between the Navigator and the Geodatabase. Since a WMS-compliant geodata server is used, the content of this interface is WMS. WMS is a standard communications protocol for map servers. It allows for use of data from several different servers, effectively creating a network of map servers from which clients can build customized maps.

*The **Map and Data Presentation Interface** is denoted by the number **6** in Figure 4-1.*



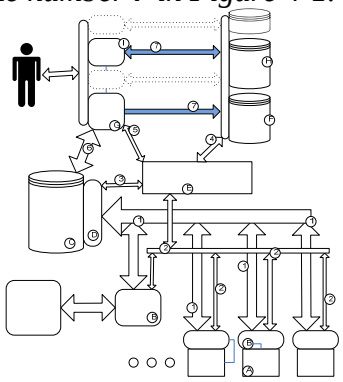
WMS servers interact with their clients by using the HTTP protocol. The WMS specification defines certain request types, and for each of those requests defines a set of query parameters and associated behaviors. A WMS-compliant server must be able to handle at least two types of requests, GetCapabilities and GetMap. A GetCapabilities request returns an XML document containing metadata about the map server. A GetMap request returns an image of a map based on the input information. Support exists for other requests such as GetFeatureInfo, DescribeLayer, and GetLegendGraphic. A GetFeatureInfo request returns information about features at a query location. This specific request type is used to get data when a user clicks on the map. The DescribeLayer request returns an XML description of one or more map layers. Lastly, the GetLegendGraphic request type returns a legend image (icon) for the requested layer including labels.

The data editor program (calibration tool) will request maps or other graphical data from the GDB. The GDB will act on that request and deliver a map (image) or other data. A fairly high bandwidth connection is needed because multiple users may be requesting maps nearly simultaneously. Importantly, when the user edits the map or enters data, the results are passed into the GDB.

In addition to communicating map layer data, this interface must support the communication of non-spatial data and a significant amount of system control. When a new run is defined, information is sent to the GDB about the run's inputs so that, if these data sets do not already exist, the GDB can generate images with default data or information extracted from another source.

4.2.7 Web Page Generator and Database Interface

*The **Web Page Generator and Database Interface** is denoted by the number **7** in Figure 4-1.*



The Web Page Generator and Database Interface governs communications between the application components (Navigator and Planner) and the relational databases (Scenario and Project and Planning). The Scenario Database will store information about each scenario. Each scenario will have one or more pages associated with it, and each page will have one or more inputs (e.g., text fields, radio buttons, checkboxes, drop-down selectors, maps) associated with it. The inputs will provide the content for each page, and each page will be a step in the scenario workflow.

The pageCreator will be an object between the Navigator user interface and the Scenario database that handles the page generation for a specific scenario. On the basis of the scenario chosen by the user, the pageCreator object will query the Scenario database for pages associated with that specific scenario. For each page, the pageCreator object will query the database for an HTML template (controlling screen layout and design) and inputs to be included on the screen. This will allow most pages of the Navigator to be dynamically generated and populated from the database.

The design styling of each Navigator screen, including placement of inputs, will be controlled by an HTML template. This template may be stored in the Scenario database and associated with a specific scenario/page, enabling customization of individual screens. The template will contain placeholder variables, which will be replaced with appropriate inputs at runtime. This model will allow for generic types of templates to be used in the future. In addition, individual input styles will be controlled by values stored in the database for each input (e.g., width, font, minimum/maximum characters, etc.).

The Project and Planning database stores information about projects, runs, and associated scenarios. The Planner user interface interfaces with the Project and Planning database in a more mundane, create-retrieve-update-delete (CRUD) fashion, than how the Navigator interfaces with the Scenario database.

4.3 BEHAVIORS AND INTERACTIONS

This section presents the principal interactions among components and, in combination, the behavior of the system as a whole, in the form of swim lane diagrams. The swim lane is a visual element used in process flow diagrams that depicts what or who is working on a particular subset of a process. The swim lane flowchart differs from other flowcharts in that processes and

decisions are grouped visually in lanes. Parallel lines divide the chart into lanes, with one lane for each person, group, or subprocess. Lanes are labeled to show how the chart is organized.

4.3.1 Project Workflow

Figure 4-9 illustrates the interaction between the Planner/Navigator, a planner (end user), the Executive, and services (model environment, models, and GDB) for an entire project.

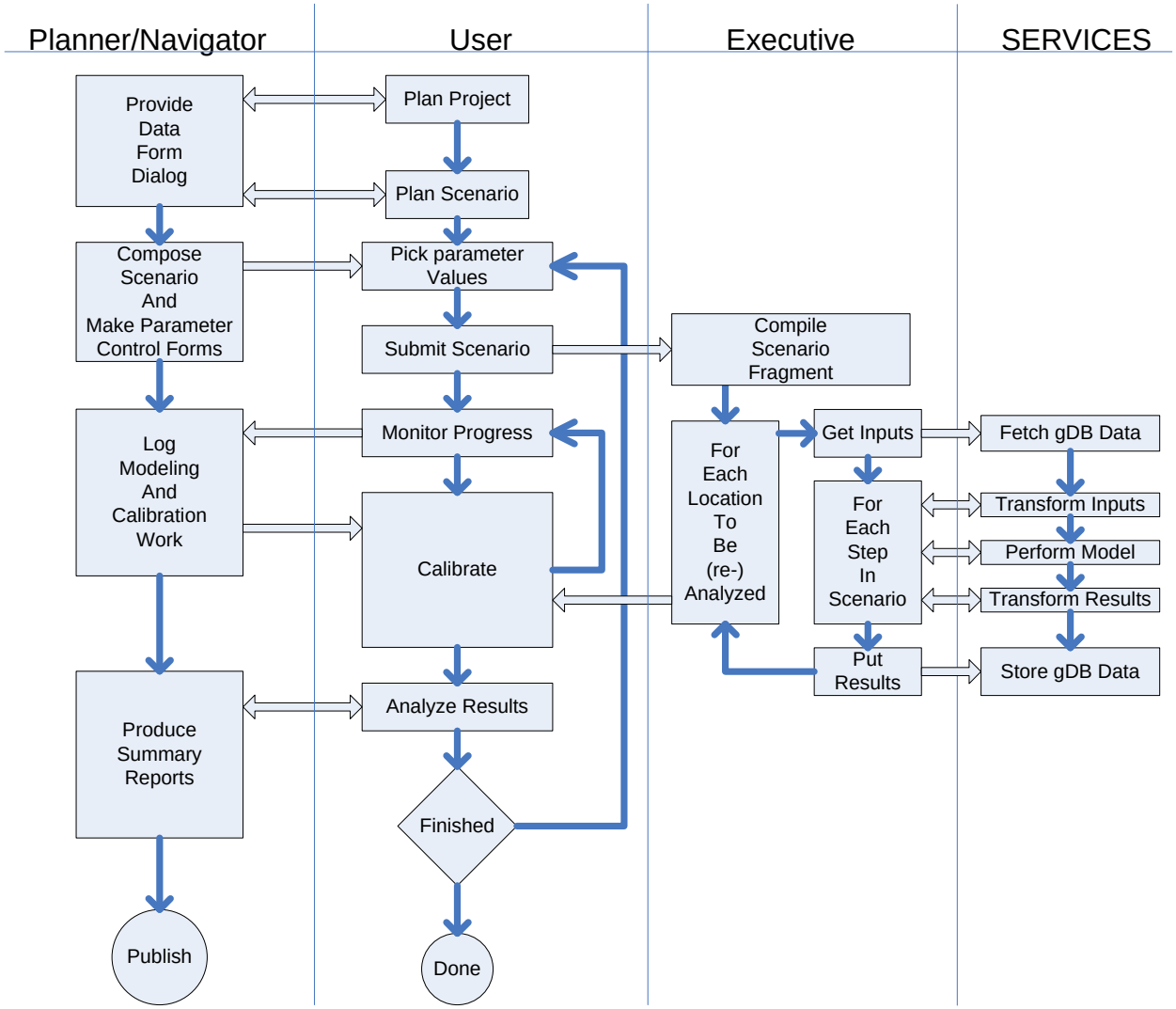


Figure 4-9. Overview of run execution.

4.3.2 Executive Scenario Compilation

The swim lane diagram in **Figure 4-10** depicts the communication between the Navigator, Executive, and Scenario DB when a scenario segment needs execution.

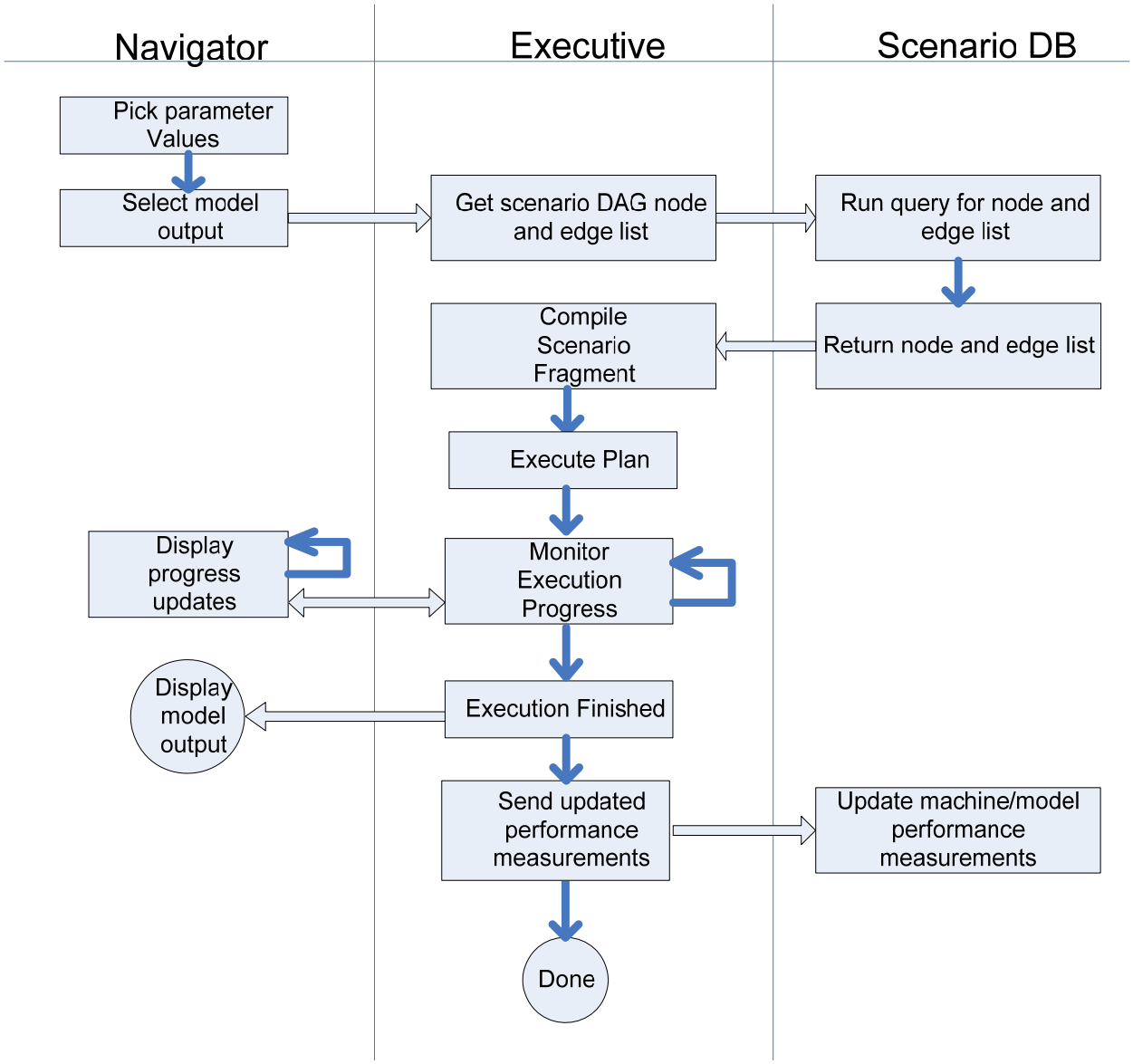


Figure 4-10. Swim lane diagram of the Navigator, Executive, and Scenario DB interaction.

4.3.3 Executive Orchestration of Model and Geodatabase Interaction

Figure 4-11 illustrates the interaction between the Executive, Data Sequencer and GDB, and models, beginning with Executive compilation of a scenario segment. This interaction occurs after user action caused the Executive to begin and is essentially a drill down into the "Execute Plan" step in the Executive swim lane in Figure 4-10.

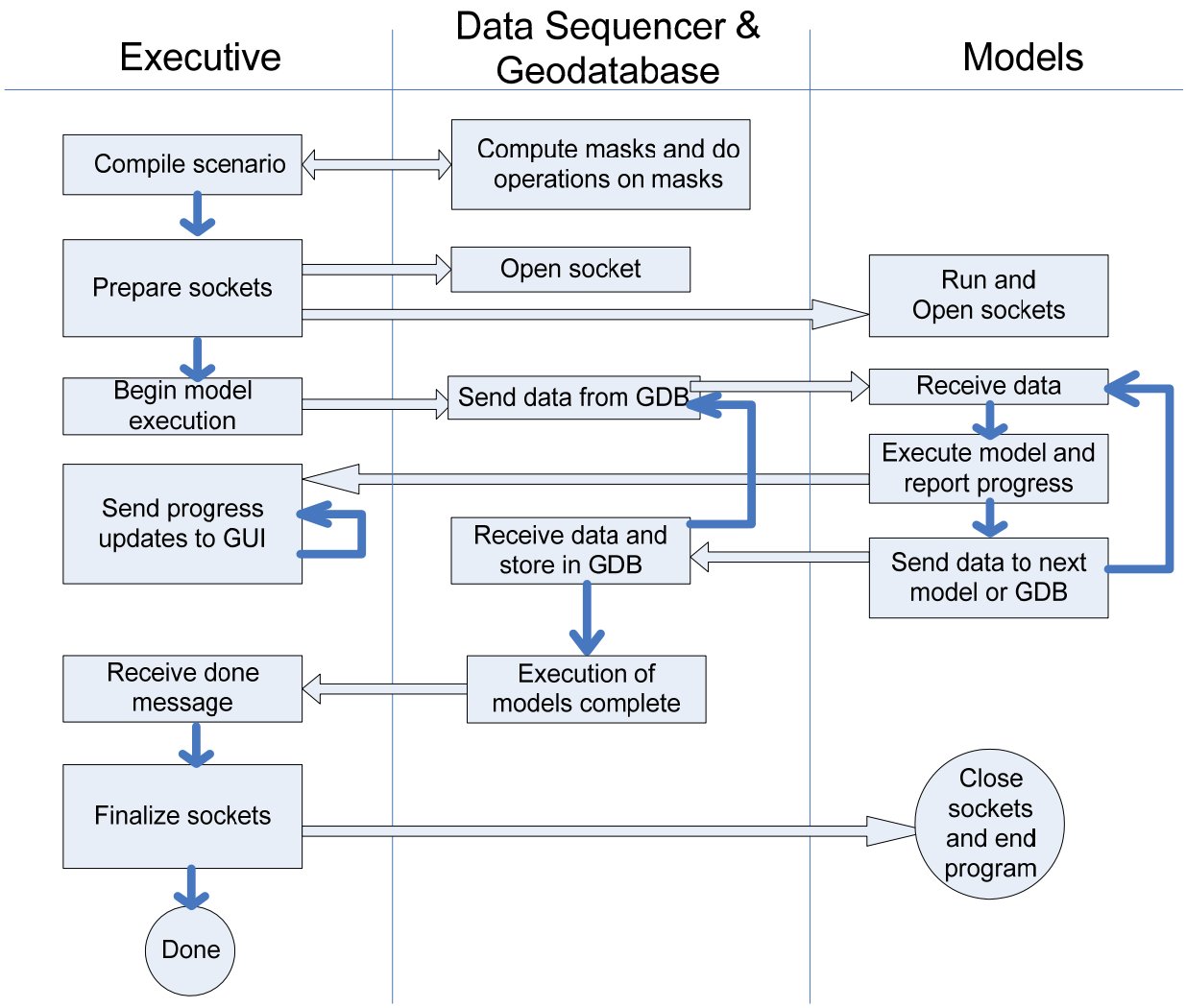


Figure 4-11. Swim lane diagram of Executive, Data Sequencer and GDB, and models interaction.

4.3.4 User Interaction with the Graphical User Interface and Database

Figure 4-12 illustrates the interaction between the user, Navigator, pageCreator object (which handles the generation of scenario specific web pages), and the database that holds the data necessary to create the web pages.

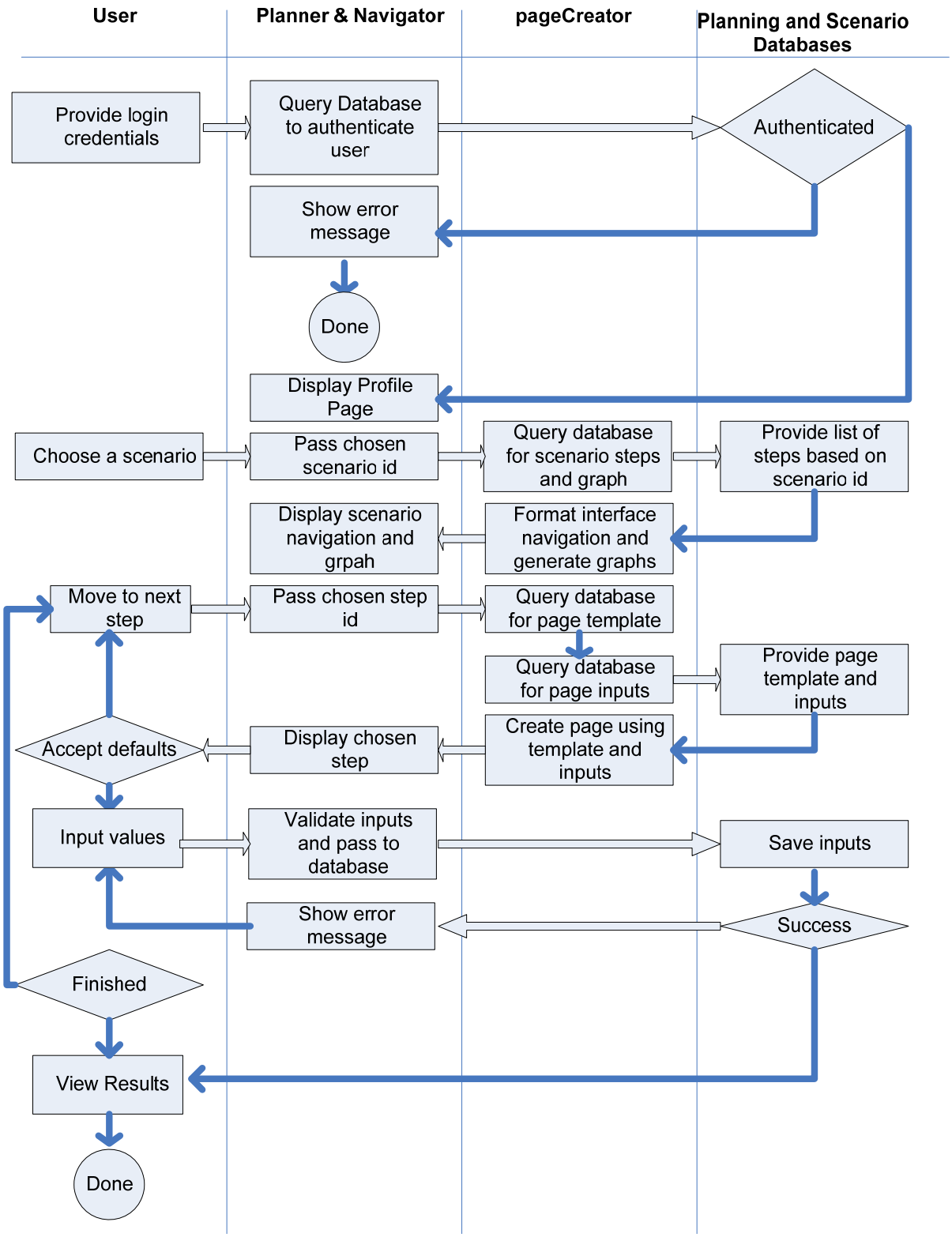


Figure 4-12. Swim lane diagram of the user, Navigator (GUI), page creator, and scenario database interaction.

4.3.5 User Interaction with the Web Map Service and the Geodatabase

Figure 4-13 shows how the User, WMS, and GDB interact with each other.

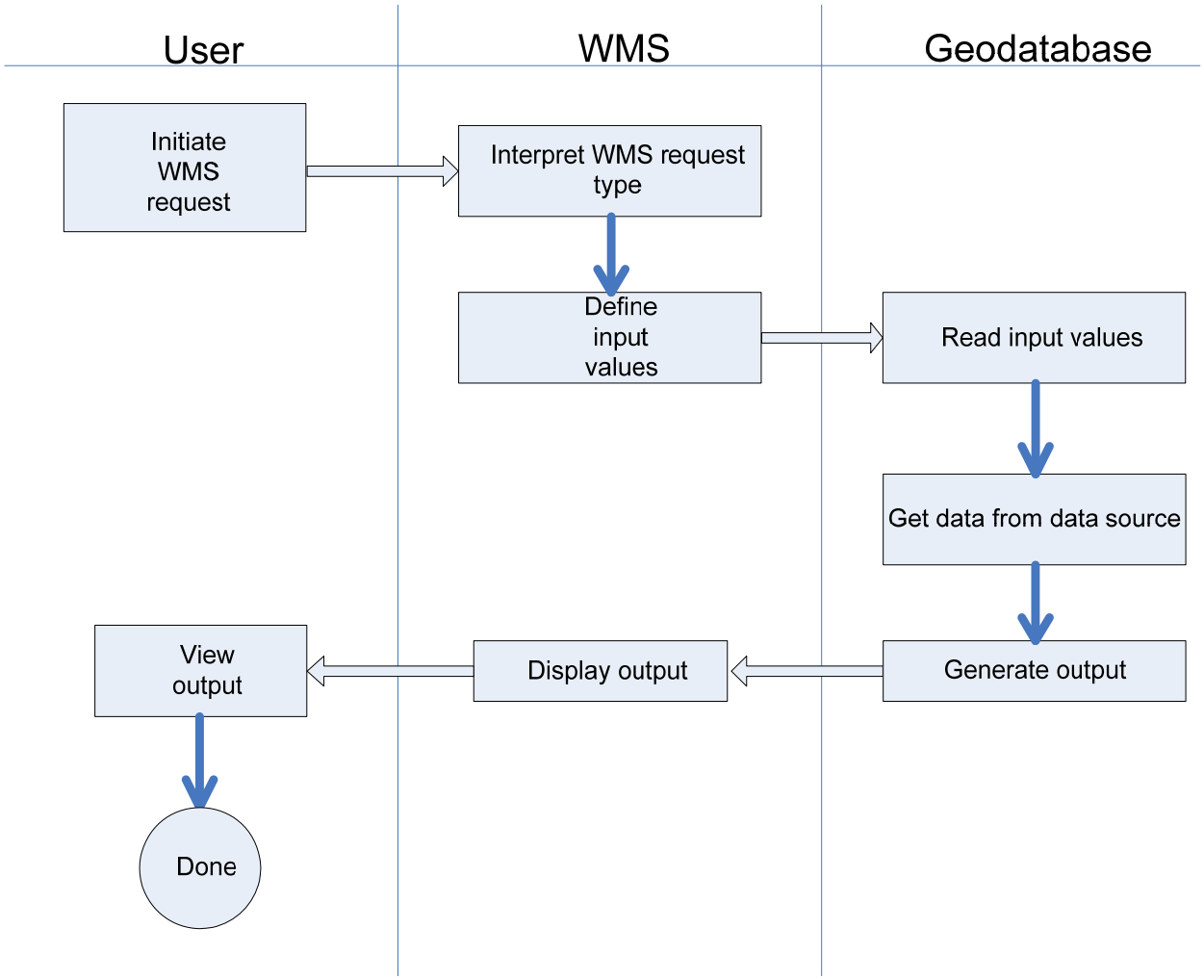


Figure 4-13. Swim lane diagram of the user, WMS, and GDB interaction.

5. TECHNOLOGIES

This section discusses specific technology choices for each component in the POC system. Flexibility in meeting the needs of the user communities is paramount to the success of the IFT-DSS. Therefore, the choices that we suggest in this section are not necessarily final. It is intended that this document be a "living document" because the design and the implementation technologies may change during development and those changes will be noted in revisions to these specifications.

Given the SOA of the system, technology decisions can be made separately for most components. Moreover, the architecture can accommodate a wide range of platforms and network topologies, from all components running on a single machine to a dedicated machine or cluster of machines for each component. Specific technology choices are based on a combination of criteria, including quality, ease of use, interoperability, reliability, performance, support, and cost.

We defer specifying network topology and many hardware requirements until we are closer to deployment, at which time we will have gained knowledge of optimal configurations.

5.1 MODELS AND MODEL ADAPTORS

The models are the components that process the parameter data. The Model Adaptors are the components that enable models to be "plugged in" and that direct parameter data into and out of the models. Models and Model Adaptors are described in sections 4.1.1 and 4.1.2. Each model employed in the system will always run in-process with, and hence on the same box as, a Model Adaptor.

<u>OPERATING SYSTEM:</u> **Windows** and **Linux**

It is incumbent on the system to accommodate models embedded in pre-existing programs that were developed to run on specific OS platforms so both of these platforms must be deployed to host the models.

<u>PROGRAMMING LANGUAGE:</u> **Java, C++**

Wrapped models (method #2 in section 4.1.2) are in pre-existing executable programs, and external models (method #3) are services, so the issue of programming language for these two methods is moot. Subclassed models (method #1 in section 4.1.2) require that a specific (object-oriented) programming language or languages be supported. We intend to support **Java** and **C++.** This means that two versions of the model adaptor host program (which make up the model parent class) shown on the left side of Figure 4-3 will be developed. The model adaptor host programs for the other two methods will be written in **Java.**

5.2 GEODATABASE

The primary function of the Geodatabase is to manage and serve the data that are generated and used by the models. Whereas a large fraction of these data are spatially distributed, the unique features of a GIS database are needed. Conversely, a large fraction of these data are aspatial. The data server will thus have to effectively combine the features of a GIS database with those of a fast data server, and be able to present both types of data. The Geodatabase is also used to manage and display the spatial data in the form of interactive maps as referenced in sections 4.1.3 and 4.1.7.

OPERATING SYSTEM: **Linux**

There exists a large quantity of open source resources that address similar problems. We also have access to similar tools that are currently in use by members of our user community (e.g., WFDSS). **Linux** is the preferred OS for these resources and tools.

GIS SOLUTION: **MapServer, OpenLayers**

We are investigating the suitability of **MapServer** in combination with **OpenLayers** as our GIS data and mapping solution. MapServer is an open source platform used to publish spatial data and interactive mapping applications to the web. It was chosen because of its many features, which include advanced cartographic output, support for popular scripting and development environments, cross-platform support, support of numerous Open Geospatial Consortium (OGC) standards, availability of a multitude of raster and vector data formats, and map projection support. OpenLayers is a pure JavaScript library used to display map data in web browsers, with no server-side dependencies.

We are also investigating how we might leverage the mapping interface from WFDSS.

5.3 DATA SEQUENCER AND EXECUTIVE

These components are the inner "clockworks" of the system. The Data Sequencer interacts with the Geodatabase to emit input data to the models and to store model output data. The Data Sequencer also provides data filtering calculation services to the Executive. The Executive is the program that compiles, and triggers execution of "runs", of scenarios.

OPERATING SYSTEM **Linux**

PROGRAMMING LANGUAGE **Java**

Linux and Java are both robust, reliable, enterprise-class technologies, and well suited for these non-GUI components. They are both zero-cost. Java is easier to "use" than C++, and Java is cross-platform, which means that Java-based components can easily be re-used for any subsequent deployments to other OS platforms.

5.4 PLANNING AND SCENARIO DATABASES

OPERATING SYSTEM **Windows Server 2003 64 bit or higher**

RDBMS **MS SQL Server 2008 64 bit**.

The Planning and Scenario Databases require a relational database management system that can scale with future growth. Four database systems were reviewed: current versions of Oracle, SQL Server, MySQL, and PostgreSQL. For IFT-DSS project success, experience with the proposed database is critical. STI has over 10 years of experience with SQL Server and Oracle systems, both operationally at STI in projects such as AIRNow (Anderson et al., 2002) and with user site installed systems such as the Data Management System (Gray et al., 2004). Microsoft SQL Server 2008 is the most cost-effective solution for both development and production systems. Running SQL Server in a 64-bit environment will allow it to take advantage of large memory resources if needed for future system features. STI has experience with SQL Server for both single machine single user projects (SQLExpress) and billion record operational databases used in SOA Web service environments.

5.5 NAVIGATOR AND PLANNER

These are the user-facing parts of the system; they present a browser-based, interactive graphical user interface. The Navigator and Planner interact with the scenario and planning databases to manage projects, work scenarios, and so on. The Navigator interacts with the Geodatabase and the Executive to view, edit, and generate new parameter data.

OPERATING SYSTEM **Windows Server 2003**

WEB SERVER **Internet Information Server (IIS)**

Windows Server 2003 with IIS is a common environment for hosting web applications. STI has extensive and positive experience developing, maintaining, and hosting web applications in this environment.

PROGRAMMING LANGUAGE **PHP, JavaScript, HTML**.

PHP is proposed as the server-side language for the following reasons: 1) PHP has become a commonly used and understood language, and this choice will facilitate future modifications of the system by various parties; 2) PHP can be run on multiple platforms and web servers, which will facilitate deployment on Linux/Apache if necessary; 3) there are no software licensing fees; 4) STI has expertise developing software using PHP and is currently building the BlueSky Playground application for the USFS using this language.

6. REFERENCES

Anderson C.B., Dye T.S., Ludewig S.A., Chan A.C., Gray E.A., and Prouty J.D. (2002) Functional design specification document for the AIRNow technical user web site. Design specification document prepared for U.S. Environmental Protection Agency Office of Air Quality Planning and Standards, Research Triangle Park, NC, by Sonoma Technology, Inc., Petaluma, CA, STI-902087-2287-DSD, December.

Drury S.A., Rauscher H.M., Raffuse S.M., and Funk T.H. (2009) Refined work flow scenarios and proposed proof of concept system functionality for the interagency fuels treatment decision support system. Final report prepared for The Joint Fire Science Program, Boise, ID by Sonoma Technology, Inc., Petaluma, CA, and [Rauscher], STI-909029.02-3655-FR, July.

Funk T.H., Corman R.G., Reed J.E., Raffuse S.M., and Wheeler N.J.M. (2009) The Interagency Fuels Treatment Decision Support System (IFT-DSS) software architecture. Software architecture design prepared for the Joint Fire Science Program, Boise, ID, by Sonoma Technology, Inc., Petaluma, CA, STI-908038.04-3565, March.

Gray E.A., Prouty J.D., Ovard C.D., and Wheeler N.J.M. (2004) Data management system for aerometric data. System documentation prepared for Bay Area Air Quality Management District, San Francisco, CA, by Sonoma Technology, Inc., Petaluma, CA, STI-903730-2602-SD, September.

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.