



# Viper Technical Overview

August 2010

## Table of Contents

1	Viper Overview.....	2
1.1	Viper Collaboration Framework.....	3
1.2	Viper Solution Framework .....	6
2	Frequently Asked Questions.....	8
2.1	What are the hardware requirements for Viper?.....	8
2.2	What operating system platform does Viper run on? .....	8
2.3	What programming language is Viper written in?.....	8
2.4	Is a Viper SDK available?.....	9
2.5	How mature is the Viper software? .....	9
2.6	How does Viper support collaboration across machines? .....	9
2.7	How does Viper support chat collaboration?.....	10
2.8	How does Viper support file collaboration? .....	10
2.9	How does Viper support whiteboarding collaboration? .....	11
2.10	How can Viper be used to integrate 3rd party plug-ins and tools?.....	13
2.11	How can Viper be used to search multiple database (SQL) sources and display from them in geospatial, graphical and tabular form? .....	14
2.12	With Viper, can a developer modify the user interface? .....	14
2.13	How does Viper support the user defining the work environment? .....	15
2.14	How does Viper support Esri/CJMTK map data products? .....	16
2.15	How does Viper support Esri/CJMTK Analysis?.....	16
2.16	Can Viper read and write to an Esri/CJMTK geo-database?.....	16
2.17	How does Viper support symbology?.....	17
2.18	What is the network performance of Viper?.....	17
2.19	How is Viper different from an application integration package like Eclipse? 17	
2.20	How does Viper support the smart client framework concept?.....	18
2.21	How does Viper support data mining?.....	19
2.22	How is Viper Extendable? .....	20
3	References.....	21

## 1 Viper Overview

Viper is a licensable software product now available from Overwatch. The Viper product serves as the foundation for providing complex, multi-faceted analysis and command and control applications in tactical and fixed-site environments. Designed using software product line concepts developed by the Carnegie Mellon Software Engineering Institute [1], Viper enables the development of common reusable components, or software core assets (e.g. a 2D map or Link Analysis Tool), that can easily be assembled into a variety of applications to suit specific mission needs. These software core assets may be user interface (UI) components or background services and may provide client-side or server side processing capabilities. Once assembled by Overwatch or customer system developers these software core assets form a composite application in a single user interface in which data and events are collaborated between application parts and user systems.

The Viper product consists of two frameworks, the Collaboration Framework and Solution Framework, and a software development kit (SDK). The architecture utilizes common

standards, is open and utilizes elements of both the Service Oriented Architecture (SOA) and Event Driven Architecture (EDA) styles. Viper provides a type of service bus that implements the publish/subscribe and model-view-controller patterns. Viper uses elements of the SOA style to define, govern and implement core service interfaces, provide service and component discovery, and implement decoupled services. Viper uses elements of the EDA style to accomplish message routing between services and application components.

Viper provides a common data object model and a shared awareness of data object lifecycles. Any data object model can be used with Viper. The default object model is derived from the Joint Consultation Command & Control Information Exchange Data Model (JC3IEDM) to facilitate interoperability between services and with external systems. Application and service components communicate via a publish/subscribe message mechanism. The publish/subscribe pattern is used because it enables the simultaneous delivery of messages to multiple destinations (many-to-many connections). The Model-View-Controller (MVC) pattern is used to separate the business logic involved in the object lifecycle from the view of the data.

An implementation of a Viper-based system consists of decoupled components that adhere to common interfaces provided in the SDK and fit within the Collaboration and



**Figure 1 - Example Viper composite application.**

Solution Frameworks. The components may be developed in-house or provided by a 3<sup>rd</sup> party (e.g. Commercial Joint Mapping Tool Kit (Esri/CJMTK)). These components can be packaged together or obtained (e.g. downloaded) separately for inclusion into the system. Adding new components to the system does not require modifications to existing components.

## 1.1 Viper Collaboration Framework

The Viper Collaboration Framework is an event-based service integration framework that provides a service bus. The service bus is a set of service containers that are interconnected with a messaging bus. Viper does not require an application server and can scale down to a single application and/or process. It is focused on building applications that are to be integrated into enterprise architectures. Systems are composed of Viper-compliant applications components and services (referred to as Viper clients) that interact via the Collaboration Framework using a publish/subscribe pattern.

Viper clients publish and subscribe:

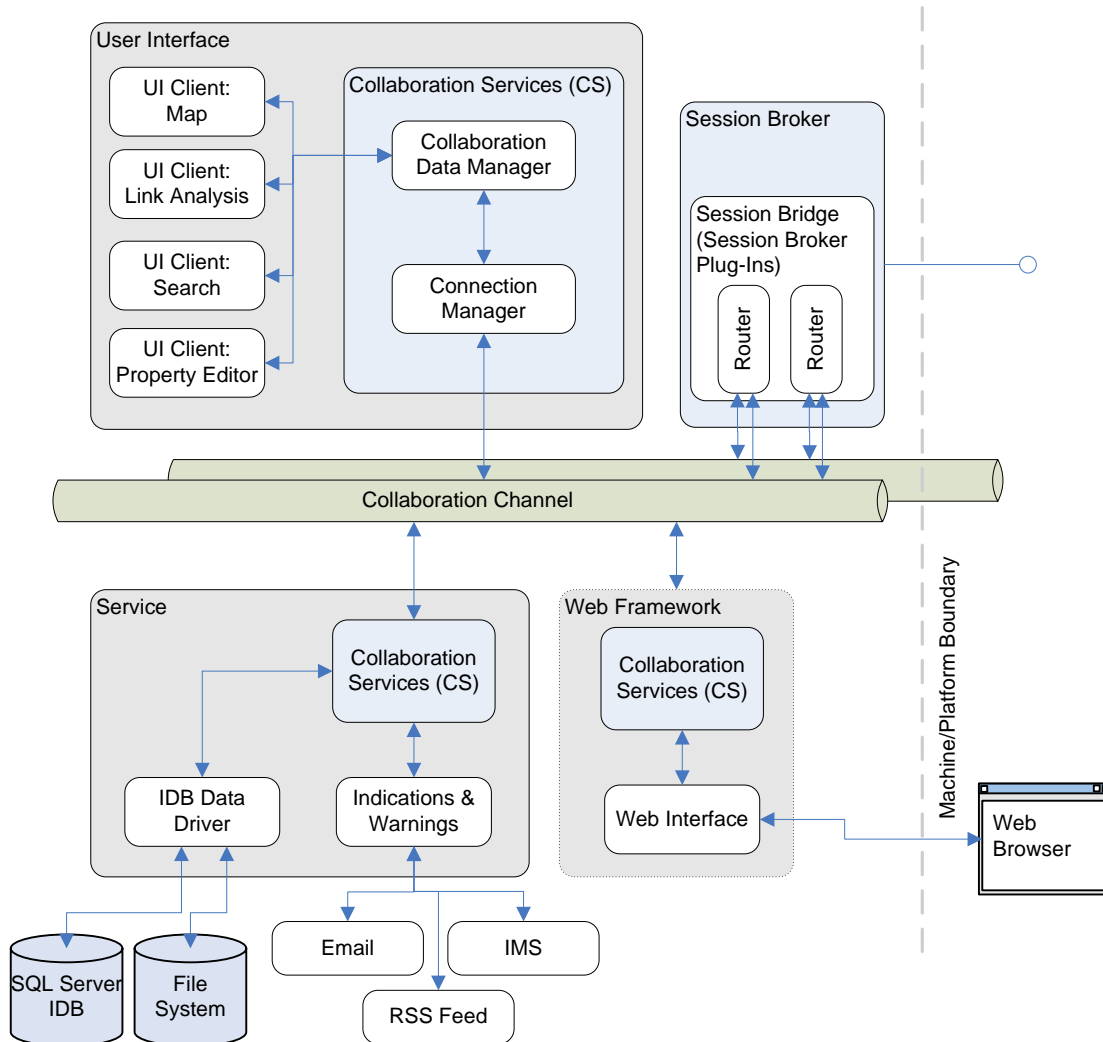
- Events, such as object creation, modification, or deletion, user actions.
- Objects, such as a person, line of bearing, track or query (objects can be added/extended as required) .
- Interfaces, which are subscriptions based on capabilities. For example, a properties editor component client can subscribe to and edit any object that implements iEditable interface. A map component client can plot any object that implements iMappable interface.

Figure 2 shows a diagram of the Viper Collaboration Framework architecture. The main elements of the architecture are:

- **Collaboration Data Manager (CDM)** – in-process object management and event notification. It handles activities such as creation and deletion of objects and client subscriptions.
- **Connection Manager** – windowed client adapter that provides a user interface for connecting to local or remote Session Brokers to manage their sessions (add or remove).
- **Session Broker** – manages available sessions on a single system. It handles activities such as keeping track of available sessions, session types and channel types, subscriptions, and session creation and deletion.
- **Session Bridge** – Allows messages to be routed between different client/server sessions, between client/server sessions and interactive collaboration sessions, and between different interactive collaboration sessions. For example, 2 client machines are connected in client/server mode to the same server. Both clients are viewing the same entity on a 2D map. Client 1 moves the entity and saves changes to the database. The commit router detects the committed change and propagates it to client 2's session. The entity moves on the 2D map on client 2's display.
- **Collaboration Channel** – enables inter-process, asynchronous messaging between components. It is essentially a multi-cast message queue. Abstracted through an IChannel interface, it can be implemented in a variety of ways, such as

## Viper Technical Overview

with existing libraries like the Microsoft Message Queue, Windows Communication Foundation and Java Messaging Service.



**Figure 2 - Viper Collaboration Framework. The dotted line around the Web Framework indicates that it is still under development.**

Viper clients publish objects and subscribe to object types or interfaces within the CDM. For instance, a sensor feed client may publish detected emitters to the CDM. Because these emitter objects have a known position, they implement the `IMappable` interface which provides the emitter's identifier, name, latitude and longitude. The map client subscribes to the `IMappable` interface. Any time the CDM receives or notices a change to an object that implements the `IMappable` interface, it would notify the map client and the map client could then render the object to the map. As the sensor client updates the position of the emitter, the map client renders the new position in real-time. Since the emitter object is also published to the Collaboration Channel for out-of-process communication, the map client does not need be running in the same process as the sensor client in order to track the position of the emitter.

## Viper Technical Overview

Viper supports interface adapters, which allows clients to receive data that they have not directly subscribed to. When the CDM determines the interfaces a received object implements, it interacts with an adapter factory, which provides any associated interface adapters. This allows the CDM to adapt the object interfaces into interfaces that its clients are subscribed to and compatible with.

Depending on system needs, portions of the architectural elements shown in Figure 2 may be included or excluded. For example, if one needed to only build a single stand-alone application that exists in a single process, then only a set of clients and a CDM can be used. If a background service, such as a correlation service is needed, then it could be added in a new process. If collaboration across system boundaries (e.g. user-to-user collaboration) or across collaboration channels (e.g. .NET remoting and web services) is needed then a Session Broker and Session Bridge may be added.

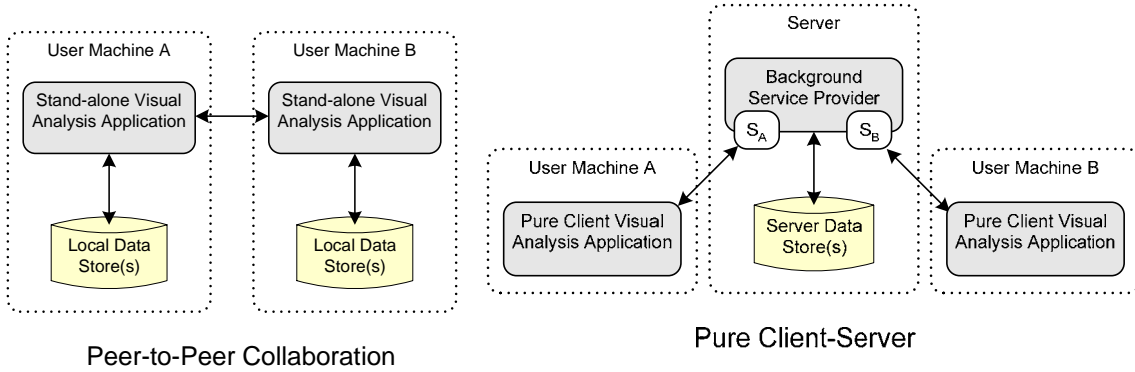
Multiple, diverse data sources and services are integrated into the system through the creation of data adapters and service clients. Data adapters provide both connection and mediation services between the data source and the Collaboration Framework. Legacy services (e.g. a common security service) and applications are integrated through the creation of service client adapters, which provide both connection and mediation services between the client and the Collaboration Framework.

The Viper data model can be modified as needed, but the default data objects are derived from the Joint Consultation Command & Control Information Exchange Data Model (JC3IEDM) to facilitate interoperability between services and with external systems. Because of the reliance on interfaces to access objects, changing the data model does not require client code to be modified.

The Collaboration Framework supports multiple types of data and user event collaboration. The collaboration of data occurs on fine grained objects of the common operating picture (COP), which can be manipulated and annotated. The shared information space is locally cached and updated in near real time for all users. The framework supports visibility and discovery of collaboration sessions (i.e. volunteer collaboration). Multiple users can interact with the collaboration session simultaneously and bandwidth is efficiently used through compression and the use of metadata.

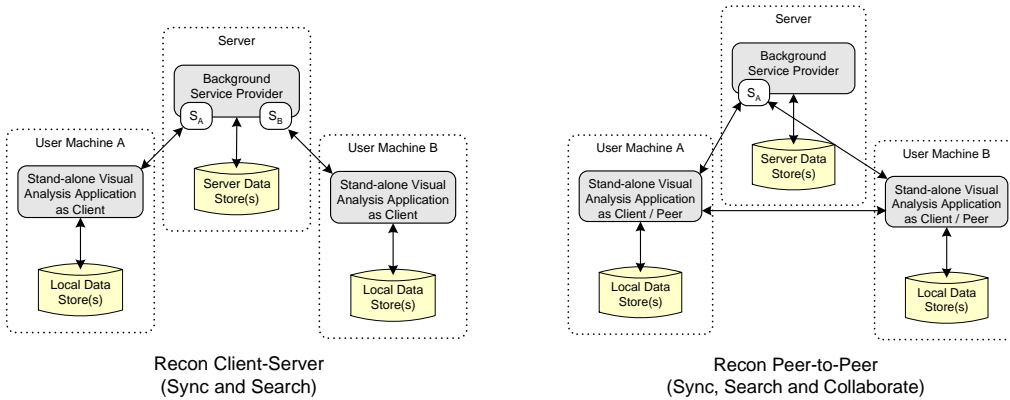
In the case of the collaboration configuration shown in Figure 3 (Left), the collaboration supports the configurable response to shared UI events, configurable synchronization with remote views, case/project collaboration, automated correlation and control over local and collaborated data modifications. In the case of the collaboration configuration shown in Figure 3 (Right), the collaboration supports a shared database and services.

## Viper Technical Overview



**Figure 3 - (Left) Peer-to-Peer collaboration between systems. In this case both data and user events are collaborated. (Right) Client-Server collaboration of data and services.**

In the case of the collaboration configuration shown in Figure 4 (Left), the collaboration supports simultaneous access to local and server databases, synchronization between local and server same-schema databases, control over local and collaborated data changes, automated correlation and shared services. This configuration is intended to allow users to work in both connected and disconnected environments. In the case of the collaboration configuration shown in Figure 4 (Right), the collaboration supports simultaneous access to local, peer, and server databases, synchronization between local, peer and server databases, automated correlation, configurable response to shared UI events, configurable synchronization with remote views, case/project collaboration and shared services<sup>1</sup>.



**Figure 4 - (Left) Client-Server collaboration between systems in which both local and server data stores are accessible. (Right) Peer-to-Peer collaboration of data and services in which both local and remote data stores are accessible. In this case, user events are also collaborated. Note: Peer-to-Peer collaboration of data and services will be available in 2007.**

### 1.2 Viper Solution Framework

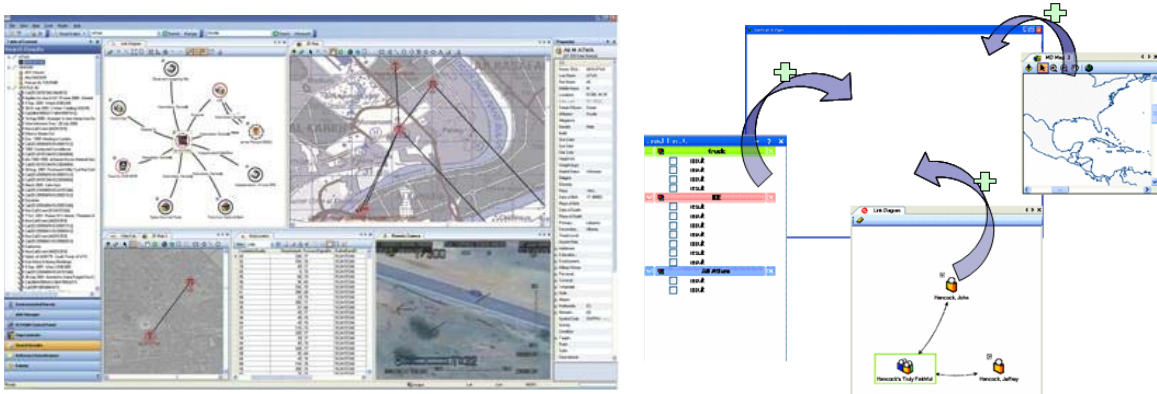
The Viper Solution Framework is an object oriented execution environment for integrating together application elements. The Solution Framework provides a set of interfaces to be implemented and performs object brokering. It provides support for connecting and configuring a group of Collaboration Framework client components that

<sup>1</sup> This type of collaboration will be available in 2007.

## Viper Technical Overview

together form an application. It also provides support for interactively modifying which client components comprise the application and how they are connected and configured. Figure 5 shows an example application built with the Solution Framework.

The Solution Framework includes an application framework, common client adapters and common user interface clients. The application framework includes a runtime environment, and classes which provides presentation support. The primary goal of the Solution Framework is to enable product engineers to quickly create applications from a diverse set of components which may be new, legacy, COTS or GOTS.

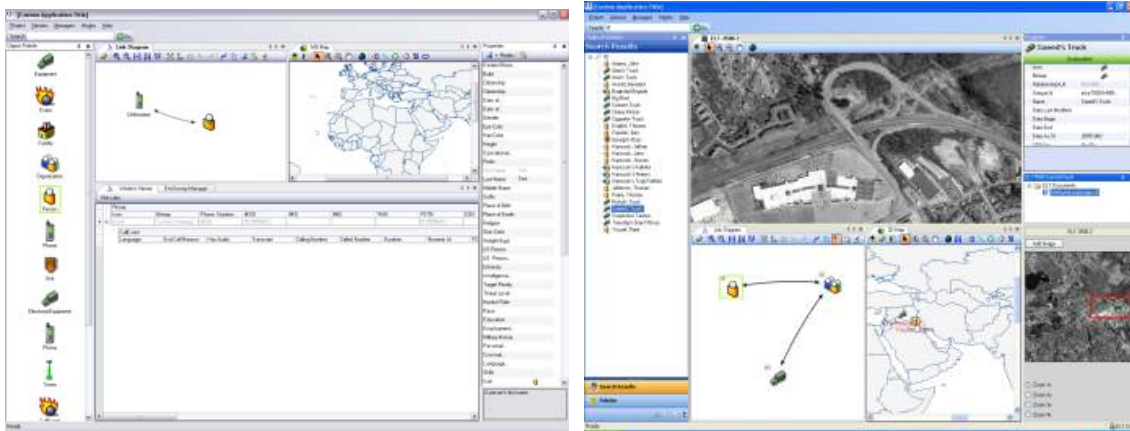


**Figure 5 - (Left) Example application built with the Viper Solution Framework. (Right) Applications can be built and easily reconfigured by adding or removing client components and configuring them. This flexibility allows the system to be easily adapted to a changing mission.**

A UI client implements various interfaces for presentation and user mode support and core interfaces for discovery, plug-in, and diagnostics support. The COTS components SandBar, SandDock and EyeFinder are used to provide dockable-window behavior. Through the use of interfaces, the Solution Framework is designed to change dockable behavior implementations without changing any of the UI client code. Background Services (no UI) implement only the core interfaces for discovery and plug-in support.

The Solution Framework encourages the pattern of dependencies through the interfaces. When building a component that provides services to other components in an application, two assemblies are created: one containing only interfaces and one containing the implementation of the interfaces. Reference to the component is only through the interfaces. Thus, the implementation of the interface could be updated without other clients needing to be modified.

## Viper Technical Overview



**Figure 6 - Example variations of an application using the Windows implementation of the Solution Framework.**

The Solution Framework improves usability and trainability in multiple ways. User roles can be defined for a single application, resulting in a variant of the application that is tailored to specific user tasks. This customization includes both what components are shown to the user and the behavior of the data in the user interface. Customization is also available on a per-user basis. The Solution Framework increases trainability because it provides single user interfaces for tasks involving integrated legacy components. For example, a single property editor may be used for all integrated components instead of a different property editor for each. This same property editor may be used for new components added to the system in the future.

## 2 Frequently Asked Questions

### 2.1 What are the hardware requirements for Viper?

Viper runs on a variety of hardware configurations. It can be used in stand-alone or networked environments and can be used with different hardware for client and server configurations. It has been used in government programs with a Panasonic CF-29 laptop (1.2GHz Intel Pentium M; 256MB DDR SDRAM 333MHz; Intel 82852/82855 GM/GME graphics controller; Hitachi DK23EA-40 40GB 4,200rpm HD). It has also been used in a server configuration with a SunFire x4100, 2x Dual Core AMD Opteron Processor 285, 2.6GHz ea, 4GB RAM, 4ea 72GB HDD (two in RAID 1, 1 hot swap, 1 general), 4ea 10/100/1000 NIC, 1ea 10/100 Service Processor NIC.

### 2.2 What operating system platform does Viper run on?

The Viper product runs on Windows 2000, XP, Server 2003 and Vista.

### 2.3 What programming language is Viper written in?

The Viper product is written in C#, using the .NET 2.0 framework. Viper application components (e.g. a Link Analysis Tool component) can be written in any language compatible with .NET, for example C#, VB.NET, C++.

## 2.4 Is a Viper SDK available?

The SDK is currently available. It includes documentation (API, FAQ and instructional), a sample application (windowed clients, data adapter, additional objects) and a Viper application wizard.

The next releases of the SDK are May 1, 2007 and July 1, 2007. These releases will include expanded documentation, an expanded sample application (Preference Panel, Query Panel, interface adapter collaboration, etc.) and tools and wizards (object builder, component selector, mode generator, subscription wizard).

## 2.5 How mature is the Viper software?

The Viper product software and SDK are available now. Viper is currently being used in the multiple U.S. government programs.

In terms of Technology Readiness Level (TRL), Viper is at TRL 8, “Actual system completed and qualified through test and demonstration.”

## 2.6 How does Viper support collaboration across machines?

The Viper Collaboration Framework provides collaborative access to domain data objects managed by a shared information bus. Viper smart clients define subscriptions to the Collaboration Framework which represent their data interests. Clients are notified of data object modifications through events published by the Collaboration Framework and execute custom logic for processing the event payload (such as updating the location of an entity plotted to a map). Presentation of these domain objects is provided through well defined interfaces implemented by the data objects or exposed through the use of interface adaptors. Visual clients interact with the objects through these interfaces and provide a client-specific view of the data. For example, a map client may interact with an object through its *IMappable* interface and render the object using the appropriate symbology. Modifying the symbol (such as changing its position) on the map directly modifies the underlying object. Multiple clients can provide simultaneous, disparate views of the object using any of its public interfaces. A modification in one view is immediately reflected in the other active views through Collaboration Framework events. These events are disseminated to all clients of the Collaboration Framework running in a single process space as well as to Collaboration Framework clients in separate processes or on remote network nodes.

A key component of this architecture is a well-defined model for capturing and disseminating system-level and user-level events. Events of interest include common data management operations for shared data (create, read, update, delete), as well user interface interaction events (select, drag and drop). Event dissemination supports collaborative interactions between data consumers and producers in a variety of deployment architectures (including client-server and peer-to-peer) and at a variety of collaboration levels.

Viper’s Collaboration Framework provides data synchronization across disparate processing and analysis nodes and supports multiple, user defined views of the common

## Viper Technical Overview

operational picture (COP). The platform supports definition and publication of shared views of the operational picture. These shared views can include a common map-based representation, link-analysis displays, or data grids. Updates to shared data are reflected in all shared views in a consistent manner. The Collaboration Framework provides mechanisms for sharing user interaction details in which individual selections, keyboard entries, and drag actions are reflected simultaneously on all collaborating nodes. This level of collaboration facilitates user-to-user communication, multi-user development of dynamic intelligence products, as well as remote training and support.

### 2.7 How does Viper support chat collaboration?

Viper includes a light-weight chat/IM client that docks within the Solution Framework. The chat client uses a native Viper chat protocol. Chat messages are routed through the Viper Collaboration Framework – supporting multi-way chat sessions. Third party chat or instant messaging solutions can be integrated into a Viper application by developing an instant messaging interoperability service which performs protocol translation between the native Viper protocol and external chat/IM protocols (IRC, AIM, Yahoo Messenger, MSN Messenger). Additionally 3<sup>rd</sup> party chat/IM clients can be embedded directly into a Viper application (assuming they expose appropriate APIs) using the Viper SDK's client adaptor infrastructure. In this architecture messaging between the clients would be brokered by external services; however the clients themselves could participate in Viper collaboration (supporting, for example, drag and drop operations between a Viper map client and an open chat session).

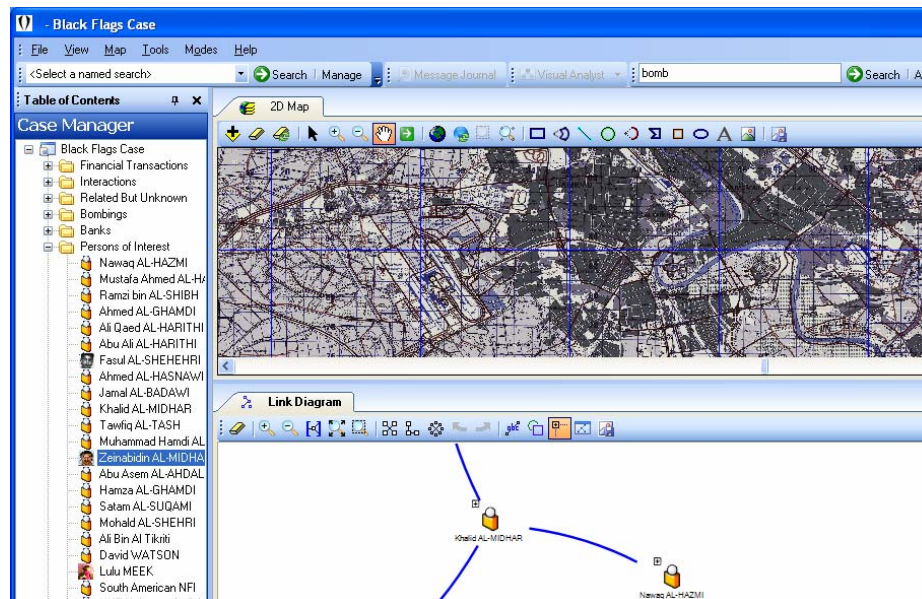


Figure 7 - Viper screen shot showing Project (Case) Manager

### 2.8 How does Viper support file collaboration?

Viper provides infrastructure for file-based collaboration through its Project Manager client. The Project Manager is a general purpose tool for organizing information (e.g.

## Viper Technical Overview

files, entities, drawings) into related *cases* using the well-understood file-folder metaphor. Cases are developed locally and can be imported and exported using an XML-based serialization of their contents. This import/export mechanism can support review and approval workflows that cross network or security boundaries. Cases can also be shared between collaborating Viper nodes – allowing users to view and modify their contents concurrently. The collaboration capabilities include automated display of visual cues indicating when information in the case has been modified.

Collaboration on the contents of 3<sup>rd</sup> party files (such as a Microsoft Word document) is provided through the relevant 3<sup>rd</sup> party toolset. These collaborations can be orchestrated through the definition of User Workflows using the Viper Workflow Engine, based upon Microsoft's Windows Workflow Foundation (WWF). User Workflows capture standard business or operational procedures that require user interaction and can prompt users to perform fine grained tasks using Viper or 3<sup>rd</sup> party tools. The Viper Workflow Engine (currently under development for delivery in CY2007) can manage task routing, task state maintenance, tools selection, and evaluation of conditional logic in the procedures. The use of Windows Workflow Foundation within Viper also allows Viper clients and services (including users) to participate in externally orchestrated WWF workflows, such as those included in Microsoft's Share Point application.

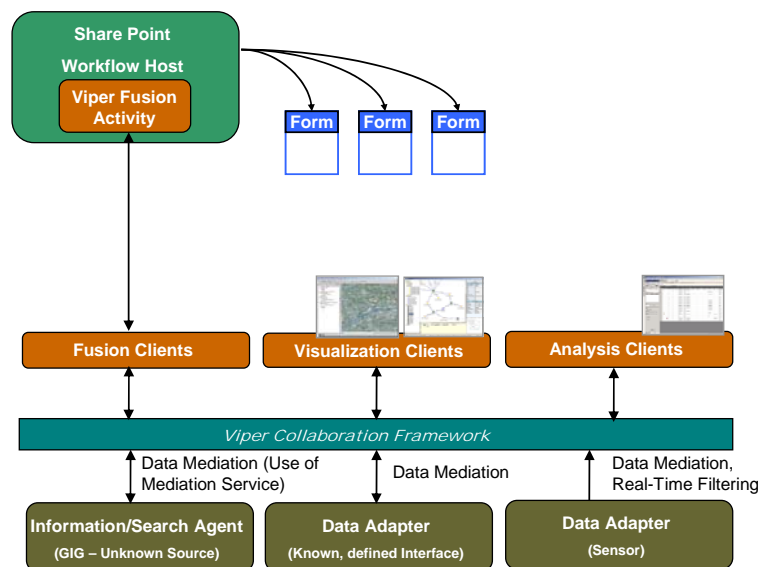


Figure 8 - Viper Services participating in externally orchestrated workflow.

## 2.9 How does Viper support whiteboarding collaboration?

As discussed in Section 2.6, Viper provides a framework for developing collaborative applications – in which data and user interaction events can be shared between Viper clients running in the same process or on remote network nodes. This allows developers and production engineers to create and configure collaborative behavior on a per-client basis (i.e., collaborative interactions between map displays might be distinct from those supported in a link analysis display). The table below describes collaborative features for existing Viper clients.

**Table 1 - Collaborative features of existing Viper clients.**

<b>Client</b>	<b>Description of Collaborative Features</b>
Project Management	Provides interface for managing arbitrary collections of battlespace entities and artifacts (e.g., multi-media INTEL products) using a file-folder paradigm. Collections of data are known as <i>cases</i> . Users can share cases to support collaborative analysis. Contents of the cases are synchronized between collaborators and modification events are indicated by a visual cue. In subsequent releases the Project Manager will support collaborative navigation of cases.
Properties Editor	Generic interface for displaying/editing object properties. Users in a collaboration session share selection events for items presented in the UI. Selecting an entity (e.g., a Person) in one Viper instance causes the entity to be selected in all collaborators and the entities properties are displayed in the property viewer. Property modifications are reflected in all collaborating Property Viewers simultaneously.
Link Analysis Tool	Integrated link-node display for Multi-INT data. Networks displayed in the Link Analysis Tool are displayed on all collaborators. Addition of entities or relationships to the diagram are reflected simultaneously. Collaboration includes expanding and collapsing portions of the network. . Separate Link Analysis Tools can display different networks, or different parts of the same network, and viewers can control the scope of their collaboration by using shared data frames to explicitly synchronize with other Link Analysis Tools.
Map	Integrated Esri MapObjects based map. C/JMTK map available in Q2 CY2007. Map collaboration allows users to share map layers and overlays. Symbols and icons plotted to the map are displayed on all clients in a collaboration. Changes to symbol/icon position (through drag and drop or manual location editing) are reflected on all clients. Tactical graphics, vector shapes and free draw elements are all shared dynamically.
Search	User interface and logic for defining search criteria against distributed data sources and displaying search results. Users in a collaborative session can share search results based upon a query constructed on any collaborating Viper node.

## 2.10 How can Viper be used to integrate 3rd party plug-ins and tools?

Viper is specifically designed for the integration of 3<sup>rd</sup> party tools. There are three methods to integrate a 3<sup>rd</sup> party tool into a Viper-based solution: service client, wrapped, and embedded web interface.

A Viper client, which is an application component that uses the Collaboration Framework, may also be a client to a 3<sup>rd</sup> party tool that provides a service interface. Typically, this type of integration is done with XML/SOAP, but other methods are also supported. For example, the Nexidia audio mining tool was integrated by creating a Viper client that provided a UI for audio mining. The Nexidia Viper client interacts with the Nexidia service interface to provide the search and processing capabilities. From the user perspective, Nexidia capabilities are part of the Viper-based application, allowing a user to drag-and-drop audio objects onto Nexidia.

3<sup>rd</sup> party software components can be integrated directly into the Solution and Collaboration Frameworks. Integration is achieved through development of a wrapper, or client adaptor, to the 3<sup>rd</sup> party component's API. The wrapper implements the required interfaces to allow the component to be loaded into the Viper-based UI and communicate via the Collaboration Framework. For example, the ELT imagery product produced by Paragon Systems (acquired by Overwatch) was integrated into a Viper-based application by creating a wrapper. The ELT imagery component is seamlessly integrated in the Viper UI. Users can drag-and-drop data objects onto the ELT imagery component and the component responds to collaborative events (e.g. data object selection).

Java software components can be integrated directly in the Viper UI using a similar wrapper method. OWS has demonstrated this type of integration with a government legacy Java application. A Viper client, in the Solution Framework process, creates a Java UI tool, which also runs in the same process. Viper client code creates a normal Viper tabbed form view panel. Viper client code finds the window that the Java application created, subclasses the window, and sets its parent to be the empty Viper panel. The Java application runs in a version of the Sun Java Virtual Machine (JVM) which can be loaded in process. For a similar effect, a third party tool like JNBridge can also be used. Interaction with the Collaboration Framework can be achieved using a variety of messaging transports.

A third method of integrating 3<sup>rd</sup> party tools is to use an embedded web interface. This method has been used to integrate with existing enterprise architectures that contain web-based interfaces to existing applications. Using this method, a wrapped browser control is used as a Viper client. The browser control uses the available web interface. The advantage to using this method is the ability to drag-and-drop data onto the web interface. As an example, consider the case where an existing web search interface is available that performs a search for a particular kind of data associated with an individual. Users are familiar with this web interface and resistant to change. Creating a Viper client with an embedded web interface allows the user to drag-and-drop an individual or set of

individuals from other Viper clients (e.g. a 2D map) to the web interface and to obtain the search results within the Viper application.

Java applets can be integrated directly into the Viper UI using a similar method. OWS has demonstrated the integration of a Java applet that is embedded in an Internet Explorer control that is also wrapped to be a Viper client. The Java applet integrates with the Collaboration Framework via a middleware tool like JNBridge or via a web service interface. From a UI perspective, the Java applet is part of the composite Viper application.

### **2.11 How can Viper be used to search multiple database (SQL) sources and display from them in geospatial, graphical and tabular form?**

Viper is specifically designed to search and interact with data from multiple data sources including databases (SQL), web services, sensor feeds, messaging systems and file based data stores (spreadsheets and log files). Using the Viper SDK, one can create multiple Data Adapters. These Data Adapters publish and subscribe data to the Collaboration Framework, interact with the data source and provide the mediation between the data's native format and the Viper object model. Data published by the Data Adapters to the Collaboration Framework is received by any Viper application component (e.g. a 2D Map, Data Grid, Link Analysis Tool) that is subscribed to that data. Subscriptions include combinations of events (e.g. object creation), objects (e.g. Individual) and interfaces (e.g. any object that implements the IMappable interface).

For example, when a user creates a search, a Query Object is published by the Search component. Multiple Data Adapters may receive the Query Object and then query multiple network database sources asynchronously. Each Data Adapter publishes the resulting data objects back through the Collaboration Framework to be accessed by subscribed components such as the Search Results, Link Analysis and 2D Map components.

Viper supports both a simple text (e.g. Google-type) search component and any number of specialized search components (e.g. temporal search). A developer can add a new search component simply by implementing the ISearch interface as specified in the SDK.

### **2.12 With Viper, can a developer modify the user interface?**

Developers can modify the UI of a Viper-based application in following ways. All of these modifications can occur without rebuilding Viper or the UI components.

1. Application UI components can be easily added or removed. Since the UI components are decoupled, removing one does not affect the others.
2. A developer can define what objects, events and interfaces the UI components subscribe to. This definition determines what data the UI will receive and what events the UI will respond to.
3. A developer can define how the data flows into the application. For example, one can define that new objects should automatically be rendered on a 2D Map or define that objects are rendered only when a user drags objects from a UI component onto the 2D Map.

## Viper Technical Overview

4. A developer or an end user can define modes or roles. A mode defines which UI components are shown and how they are arranged. For example, an application may have an analyst mode, a collector mode and a linguist mode. The Viper-based application can provide the variant of tools and configuration that best suits the user's tasks. This type of customization can be done for all users or on a per-user basis. Typically, there are several default modes provided with a Viper-based application and end-users create new modes for further customization.

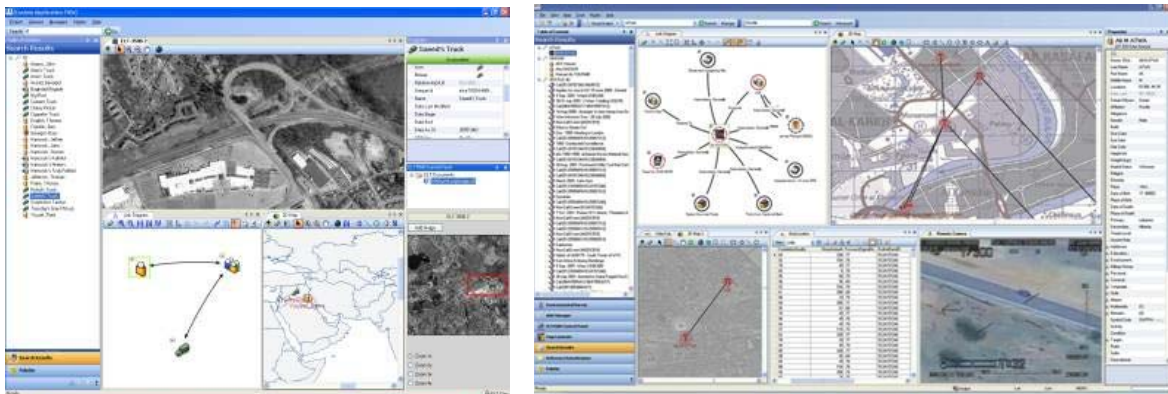
Lastly, a developer can modify the dockable-window behavior. The COTS components SandBar, SandDock and EyeFinder are used to provide dockable-window behavior. Through the use of interfaces, the Solution Framework is designed to change dockable behavior implementations without changing any of the UI component code.

### 2.13 How does Viper support the user defining the work environment?

Viper allows the user to define the environment in several ways including multiple views of the data, user modes and preferences. Using an application built with Viper, a user can use multiple views of the data simultaneously. These views may include:

- Multiple views with the same component – For example, multiple 2D maps open at the same time.
- Views of the data in different components – For example, a 2D map and a Link Analysis Tool.
- Combined views – For example, link analysis overlaid on the 2D map.
- Collaborated and non-collaborated views – the user may view data in a collaboration session in one set of views and view local, non-collaborated data in another set of views.

The types of views available are only limited by the number of components that have been developed. Views that are currently available include tabular data (grid), map, imagery, link analysis, audio analysis and many others. These views may be arranged on a single screen or across multiple screens.



**Figure 9 - Example composite applications built with Viper. The components of the application are decoupled, which allows simple addition or removal of core assets from the application.**

User modes allow for the application developer and the end user to define which components, and how many instances of a component are shown to the user, and how the components are arranged in the application. In the current version of Viper, the application developer can also use a tool that allows one to define what data the components subscribe to in a mode. A future version of Viper will provide this tool to the end user. Default modes can be delivered with the application and individual end-users can create new modes on demand. Modes can be made available to all users or single end users.

#### **2.14 How does Viper support Esri/CJMTK map data products?**

Viper currently includes an integrated Esri MapObjects-based map. A C/JMTK-based map integrated via the ArcObjects SDK (version 9.2) is under development and will be available in Q2 CY2007. The list of supported map data products for MapObjects and ArcObjects can be found at the ESRI web site (<http://www.esri.com/products.html>).

#### **2.15 How does Viper support Esri/CJMTK Analysis?**

The integrated MapObjects and ArcObjects-based maps support common C/JMTK map analysis functions:

- Load and manage map layers.
- Create, import, edit, persist and export map overlays.
- Render standard and custom symbology (e.g. 2525B).
- Create and modify geometric shapes (rectangles, circles, polyline and polygons), free draw.
- Create and modify tactical graphics.
- Graphical definition and management of Areas of Interest (AOIs).
- Supports LAT/LNG and MGRS coordinate systems and basic mensuration capabilities.

The ArcObjects-based map can support development of custom renderers and collaborates with other Viper clients to support drag and drop interoperability such as rendering of georeferenced communications networks, display of track history, and near-real time tracking. A Viper interface to the C/JMTK ArcMap application has also been developed. This allows Viper clients and services to plot to an external instance of ArcMap running on the same machine. This integration allows users to take advantage of C/JMTK map extensions such as Military Analyst.

#### **2.16 Can Viper read and write to an Esri/ CJMTK geo-database?**

The ArcObjects-based map will include support for performing read/write operations on the C/JMTK ArcSDE. Map independent access to ArcSDE will be provided in a future version of the Viper SDK. This will allow users to browse the ArcSDE, load map layer data, and display the data in a variety of interfaces – including imagery analysis applications such as Overwatch's RemoteView and ELT products.

### **2.17 How does Viper support symbology?**

Viper mapping capabilities include a native 2525B symbology renderer as well as a symbol palette control that supports customization of symbol labeling, preview of symbol layout, favorites lists and drag and drop interaction with the MapObjects and ArcObjects-based maps. The use of a native renderer supports plotting of symbols to a variety of Viper map and imaging clients. The MOLE 2525B renderer can also be supported with the ArcObjects map.

### **2.18 What is the network performance of Viper?**

As described in Section 2.6 the Viper Collaboration Framework supports multiple deployment architectures – including standalone, peer-to-peer, and client-server. In a client-server architecture Viper clients share common services for data persistence and retrieval and can participate in collaboration sessions with one or more client nodes. Viper client and services interact, primarily, by subscribing to and interacting with shared data objects – rather than through direct messaging. Collaboration over a network is provided by publishing data object and user interaction events to subscribing Collaboration Framework nodes. This limits network traffic to the exchange of required data objects and user event descriptions. These events are marshaled between processes and network nodes using a custom XML serialization with compression transported via TCP/IP.

Performance testing performed by Overwatch shows client notifications for data object and user interaction events on a single Viper node being received in the sub-millisecond range. Notifications sent between collaborating nodes on a high bandwidth network occur in 10-20 milliseconds with display latencies associated with updating a visualization client, such as a map, on the collaborating node of approximately 50-100 milliseconds. Collaboration is supported over low bandwidth connections (< 10000 bps) however total display latencies can reach the 5-10 second range depending on the type of data and visualization client.

Depending on the duration and frequency of interruptions, intermittent connectivity can negatively impact Viper collaboration. Short duration interruptions can be handled by the underlying transport. Larger interruptions result in termination of a Viper collaboration session. However, collaboration sessions can be reestablished by the user and are automatically re-synchronized.

Overwatch is currently investigating a number of approaches to improve the performance and robustness of the Viper Collaboration Framework. This includes addition of more performant transports and data serializers and optimization of information exchange.

### **2.19 How is Viper different from an application integration package like Eclipse?**

In many respects, such as the ability to provide a plug-in type framework for composite applications, Viper and Eclipse have similar goals. Viper is tailored for intelligence analysis and command and control applications. It includes a large number of interfaces,

basic clients and a default object model tailored for this purpose. Viper also provides capabilities for system-to-system collaboration.

## 2.20 How does Viper support the smart client framework concept?

Viper supports the Smart Client paradigm of combining and balancing the benefits of a rich client application with the deployment and manageability strengths of a thin client application. Viper provides the characteristics of a Smart Client in the following way:

1. **Makes use of local resources** – An application built with Viper is installed as a rich client on the user's system. Thus, it has access to all available local system resources to perform processing and take advantage of any installed hardware (e.g. a video camera). The Solution Framework allows one to provide a rich and responsive user experience including drag-and-drop and dockable window behaviors. The Solution Framework includes a basic set of application components including a Project/Case Manager, Properties Editor, Preferences Editor, Search, Search Results, Tree View, New Entity Palette, Session Manager, Map Contents Directory, Video Player, and 2D Map. A wide variety of more advanced application components have also been developed using Viper including Link Analysis, Text Extraction, specialized sortable data grids and numerous analysis tools. The application components, assembled together within the Viper frameworks provide a rich, powerful user experience.
2. **Makes use of network resources** – Through the Viper Collaboration Framework, applications can access any available data source or service that is local, on the Internet or on an enterprise network. The Viper SDK is used to create multiple Data Adapters and Service Clients for an application. These Data Adapters and Service Clients publish and subscribe to objects and events through the Collaboration Framework and also communicate with local or network data sources or services. For example, when a user creates a search, a Query Object is published. Multiple Data Adapters may receive the Query Object and then query multiple network data sources asynchronously. Each Data Adapter publishes the resulting data objects back through the Collaboration Framework to be accessed by subscribed search result and analysis components. Services such as a Correlation Service or Aggregation Service may be integrated as Viper components or through a Viper interface to a workflow manager.

To support web browser-based applications, Overwatch is developing the Viper Web Framework, which complements and is integrated with the Collaboration Framework. The Viper Web Framework is a thin client (browser-based) collaborative visualization framework and SDK – analogous to Solution Framework. Based on AJAX (Asynchronous JavaScript and XML), it will enable developers to build rich, interactive browser-based applications that support collaboration. It provides HTTP-based collaboration for communication with browser-based applications and provides collaborative interaction between a combination of thin-client and thick client applications. It also provides infrastructure support for web application UI, UI workflows, security (authorization, authentication), exception management, modularity and deployment.

3. **Supports occasionally connected users** – A Viper based system can operate on a single disconnected system or in peer-to-peer or client-server networks. From a user perspective, the only differences are the data sources available and the user-to-user collaboration features available. Data can be cached locally for disconnected operations. For example, when connected to a network a Viper system may be a client to a server database and not use a local database. Upon disconnecting, the local database is used. Synchronization of the data can be done interactively through the transfer of “cases,” which are aggregations of different data types, or automatically through database synchronization.

**Provide intelligent installation and update** – The initial Viper installation on a system is performed via an installer program. Viper application components are automatically discovered and loaded. Component updates or new components can be downloaded from the network (e.g. a web page) and added to the system by a simple copy procedure. Application components are strongly named to remove potential conflicts. Although the update process is not currently automated, it could be with the addition of a simple install client and a service interface to a component catalog.

## **2.21 How does Viper support data mining?**

The Viper SDK includes advanced, extensible federated search capabilities with both keyword and attribute-based searching. Multiple, diverse data sources can be integrated into the system through the creation of data adapters and the use of the Viper data access layer. These data adapters provide both connection and mediation services between the data source and Viper clients and services. Integrated Link Analysis, Map, and Data Grid views support interactive analysis of search results. In addition a number of data mining application and services are being developed for Viper to meet requirements of Overwatch customers. Overwatch’s product line approach assures that these components can be reused to meet the requirements of CVAF applications. A brief description of some of these efforts is provided below.

### **Text Extraction**

Overwatch is currently developing a Viper-based intelligence analysis application that provides assisted extraction of identifiers from collection reports. The extracted identifiers can be exported (along with extracted entity information and human network descriptions) to facilitate targeted collection. Specific, key capabilities include automatic extraction of identifiers from text documents, improved correlation of extracted entities to entities resident in the entity database, and the ability to recognize standard unstructured and semi-structured report formats. An objective goal is to provide the ability to perform extraction against a large document set in an unattended manner (batch extraction).



## Viper Technical Overview

- Work with Viper user modes
- Use the Viper data model and IInterfaceAdapter
- Create a Viper data driver
- Define new data objects using the object model builder
- Extend the preferences manager
- Add table of contents sections
- Customize generic client adapters
- Interface-based design
- Class finder
- Fine-grained assemblies

By following the guidelines for Viper component development, new capabilities can be easily developed to interoperate both with other components in the same GUI and with GUI instances on other machines.

Using the object model builder, interface adapters, various cross-cutting interfaces (e.g. IReliable, IMappable), and by creating new data drivers, standard Viper components like palette, property editor and search can be automatically extended to support the new data sources and object types.

### 3 References

[1] P. Clements and L. M. Northrop, Software Product Lines: Practices and Patterns. Addison-Wesley, 2001.