



Applications of Viper

August 2010

OVERWATCH

5301 Southwest Parkway | Austin, TX 78735 | P: 512.358.2600 | F: 512.358.2602

Copyright © 2010 All Rights Reserved

OVERWATCH™ SYSTEMS., 5301 Southwest Parkway., Austin, TX 78735, USA

POINT OF CONTACT:

Name: Jon Percy

Location: Austin, Texas

Email Address: jpercy@overwatch.textron.com

Phone Number: 512.358.2600

Table of Contents

1. EXECUTIVE SUMMARY	1
2. VIPER	1
3. SOFTWARE PRODUCT LINES.....	2
4. COMPOSITE APPLICATIONS.....	3
5. SMART CLIENTS	5
6. NETWORK DEPLOYMENT CONFIGURATIONS	6
6.1 Standalone Configuration.....	6
6.1.1 Simple Standalone	7
6.1.2 Standalone with Remote Network Services.....	7
6.2 Client/Server Configuration	7
6.3 Viper Client-Server Roaming Client Configuration	8
7. SPECIALIZED ENVIRONMENTS	8
7.1 High Volume Data Streams	8
7.2 High Latency.....	10
7.3 Intermittent Connectivity	11
7.4 Thin Clients.....	12
7.4.1 Near Term.....	13
7.4.2 Web Framework	13
7.5 Mixed Platform.....	14
7.5.1 Integration via Web Services.....	14
7.5.2 Direct Integration of Java	15
7.6 Protection Level 3 (PL3)	15
7.7 MLS	16
7.8 Safety Critical	16
8. CONCLUSION	17
9. REFERENCES.....	17

1. Executive Summary

Viper is a software framework used to create multi-faceted analysis and command and control applications in a variety of deployment scenarios. This document discusses the different uses of Viper in a variety of contexts, environments and deployment scenarios. The intent of the authors is to provide information about what is possible with Viper and to provide guidance as to the best utilization of the Viper product.

Viper implements a product line architecture and serves as the foundation of the Overwatch Intelligence Center (OIC) software product line (SPL). The OIC SPL is used to produce a family of products which are composite applications focused on the Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance (C4ISR) domain. Variation points within the SPL architecture allow the Viper framework to be applied in a variety of specialized environments and deployment architectures. Viper can be used as the foundation architecture for a product line, to create composite applications and to create smart applications. Viper-based systems can be deployed in several different network configurations including standalone, client/server and a combination of both. Viper can also be used in specialized environments such as those with high volume data streams, high latency, intermittent connectivity, thin-client requirements, mixed technology platforms, Protection Level 3 (PL3) requirements, multi-level security requirements and safety critical requirements.

2. Viper

Viper is a 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, 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 which data and events are shared between application components and user systems and accessed through a single user interface.



Figure 1 - Example Viper composite application.

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

OVERWATCH

Applications of Viper

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, but the Viper Software Development Kit includes a default object model. This default 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 these data objects as well as events via a publish/subscribe message mechanism. The publish/subscribe pattern enables the simultaneous delivery of messages to multiple destinations (many-to-many connections). Another pattern prevalent in Viper is the Model-View-Controller (MVC) pattern, which separates 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 Solution Frameworks. These components can be packaged together or obtained (e.g. downloaded) separately for inclusion in the system and adding new components to the system does not require modifications of existing components.

3. Software Product Lines

A software product line consists of a family of software systems that have some common functionality and some variable functionality. To take advantage of the common functionality, reusable assets (referred to as core assets) are developed, which can be reused by members of the family [1]. The purpose of a software product line is to develop a series of software systems in a predictable manner with predictable qualities such as rapid time to market, low defect rates and low cost. Software product lines and their benefits are discussed in detail in publications by Clements [1] [4], Hanratty [2], Jones [3], Bergey [5] [6], and Campbell [7].

A critical element of a successful software product line is the product line architecture, which must support the commonality and variability requirements of the family of software systems and support the systematic production requirements associated with building the series of software systems. The architecture must strike the appropriate balance between quality attributes such as performance, extensibility, configurability, and modifiability as dictated by the requirements of all the product line member systems. Furthermore, the architecture must account for the requirements of the production capability to support the efficient construction of the product line member systems.

The Viper product line architecture serves as the foundation for providing complex, multi-faceted analysis and command and control applications in tactical and fixed-site environments. Designed from the ground up to be a product line architecture, Viper enables the development of common reusable 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 which data and events are shared between application components and user systems and accessed through a single user interface. By following the guidelines (i.e. the processes described in the Viper Developer Primer) 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

OVERWATCH

Applications of Viper

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-based software assets like an entity palette, property editor and search can be automatically extended to support the new data sources and object types.

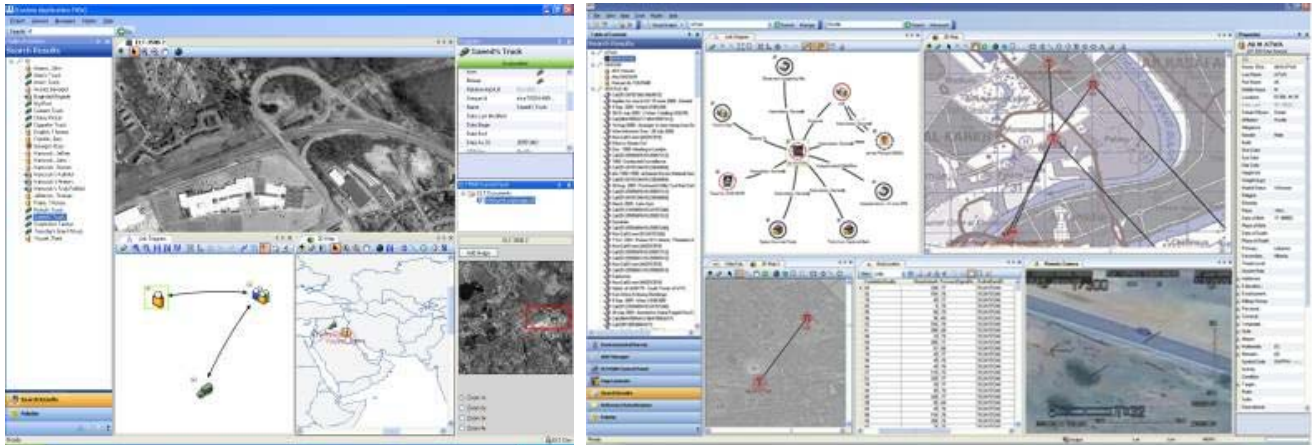


Figure 2 - Example composite applications built with the Overwatch Intelligence Center and the Viper architecture. By combining and configuring different combinations of core assets, a large number of system variants can be produced.

Viper can be used as the basis of software product lines beyond the Overwatch Intelligence Center. A customer may use Viper and the Overwatch Intelligence Center assets as a baseline on which to build a series of specialized domain applications or simply use Viper as the product line architecture, which supports a customer-developed repository of software assets.

4. Composite Applications

Although there have been many definitions proposed for composite applications, perhaps the most straight forward definition is that they are applications created by loosely coupling several different services and data stores via standardized message layers. The component parts of a composite application can be mixed and matched, much like LEGO® blocks, allowing developers to create a wide variety of applications. Similar to web-oriented mashups, composite applications combine and configure existing software assets and enterprise data sources into new applications to solve specific problems. Viper can be used as a platform to create composite applications. Fundamentally, Viper provides an application platform with which one can build a variety of applications using a mixture of prebuilt

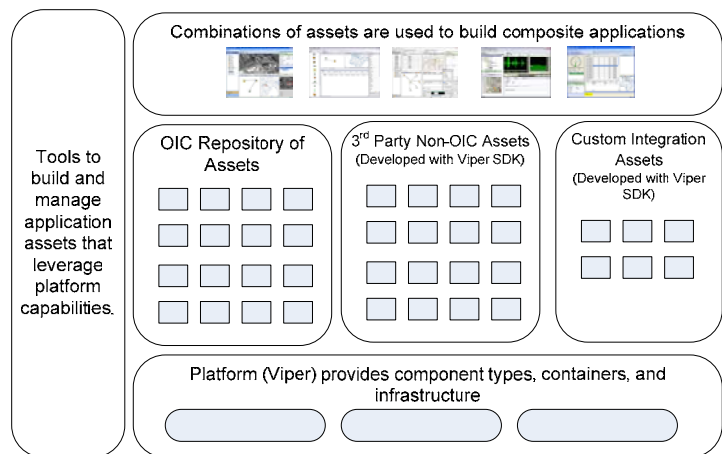


Figure 3 – Conceptual diagram of Viper and the OIC in the context of composite applications. Viper provides the common platform for the applications. The OIC provides composite assets. The Viper SDK includes tools to combine the composite assets and form the new applications.

OVERWATCH

Applications of Viper

software assets, new components and integration components. In this sense, Viper is primarily used for constructing desktop-oriented composite applications, built by combining multiple services.

Composition is combining software assets to create an application to solve a mission problem. According to Chris Keyser, lead architect for Microsoft's Global Independent Software Vendor (ISV) team, composition is a "... shift in computing from brittle, monolithic, developer-centric applications solving one particular problem, to agile, contextual, user-driven applications to support a particular business process. [8]" As stated by Atanu Banerjee, an architect in the Microsoft Architecture Strategy Team, "composition refers to a way of delivering enterprise solutions by assembling them from prebuilt components, instead of building them from scratch. It also includes personalization and customization abilities, so that users can easily and quickly modify specific functionality in the solution [9]." Viper supports composition in several ways. Using the Viper SDK, components can be created for subsequent use in multiple applications. The components are decoupled, which means that they can easily be added or removed from a Viper application without affecting other components. The components communicate with one another through the Viper Collaboration Framework, which allows them to function as a cohesive single application. The Viper Solution Framework allows multiple application "modes" to be created, which allows the application to be configured and customized to user missions or tastes. For example, an application can have an "analyst" mode and a "collector" mode.

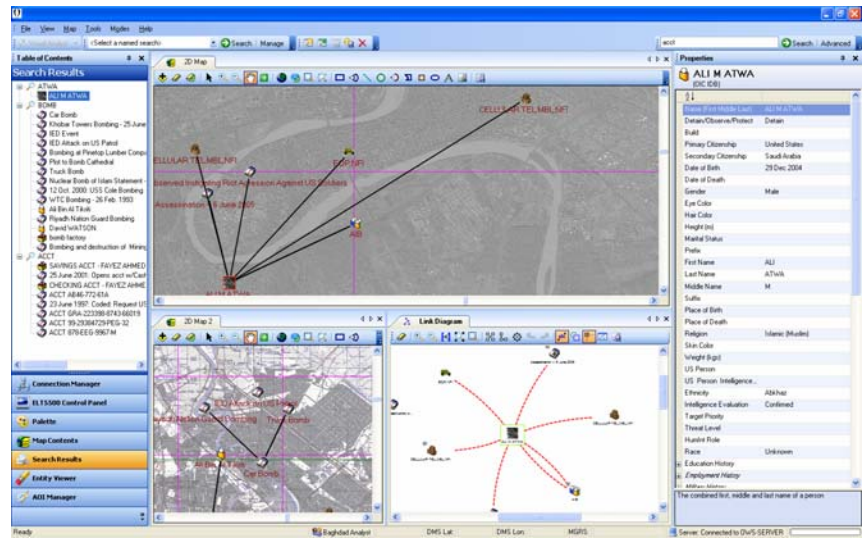


Figure 4 – Example composite application built with Viper. Composite assets shown include the Link Analysis Tool, the 2D Map and video display.

A composite asset is a discrete object with standalone value and functionality which can be combined with other objects to create an application of greater value than the sum of the individual objects [9]. In the context of the Overwatch Intelligence Center, a composite asset is a subset of the total core assets. Examples of composite assets include the 2D Map, the Link Analysis Tool, the Text Extraction component, the Interop service and the Integrated Database (IDB). Viper serves as the architecture that allows one to combine these composite assets. As one can tell from the composite asset examples, composite applications go beyond the assembly of a single user interface and include elements at all application tiers.

The architectural tiers of a composite application include the presentation tier, the service tier and the data tier. Viper provides application support for all three of these tiers. At the presentation tier, Viper allows one to combine composite assets into a single application. Viper goes beyond the simple loading of composite assets into the same user interface, and allows the assets to interact with each other through a common service bus. This interaction includes event notifications, publish/subscribe and user initiated actions such as drag-and-drop. At the service tier, Viper supports services running as background

processes which interact with assets at the presentation tier. These services can be orchestrated through the use of a workflow management system such as Windows Workflow Foundation. At the data tier, Viper supports the creation of data adapters which allow for the simultaneous access to multiple, diverse data sources. These data adapters work asynchronously in response to the data needs of elements of the presentation and service tiers.

5. Smart Clients

A smart client is an application that attempts to bridge the gap between web applications and desktop applications. In a smart client, aspects of web applications such as remote access to data, and aspects of desktop applications such as high performance and high productivity are combined. 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 communications 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, or on a remote network, such as the Internet or an enterprise network. The Viper SDK is used to create data adapters and service clients for an application. These data adapters and service clients publish and subscribe objects and events through the Collaboration Framework and communicate with local or remote 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 access, the Overwatch Intelligence Center provides a simple web application to create, view, edit and delete data entities.

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.

4. **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. 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.

6. Network Deployment Configurations

The network deployment configuration describes how a single or multiple Viper-based system(s) may operate in a particular environment. Viper has been designed to provide substantial deployment flexibility ranging from standalone operation to client-server operation. The following configurations are discussed here: simple standalone, standalone with remote network services, client/server, and client/server – roaming client.

To understand the possible network deployment configurations, it is necessary to understand some of the key components of Viper. The Collaboration Data Manager (CDM) is the Viper component that Viper application client adapters use to share data within a Viper composite application and that Viper applications use to share data in a client/server configuration. It is this usage that plays a role in defining Viper’s network deployment configurations.

Client adapters are one of the types of components that make up Viper composite applications and are the components that connect to the CDM. There are two types of client adapters:

1. Windowed Client Adapters – these are the client adapters that are directly visible to the user, such as the Viper map.
2. Windowless Client Adapters – these are client adapters that perform behind the scenes operations such as Viper’s normalization component. There are three variations of Windowless Client Adapters:
 - a. Normal Windowless Client Adapters, which are loaded in the same way that the Windowed Client Adapters are loaded by the framework.
 - b. Background Process Windowless Client Adapters, which are loaded by a special Background Process Loader component. Background process client adapters are loaded differently depending on the network deployment configuration. In a standalone Viper configuration, background process components are loaded in-process locally. In client/server Viper configurations, background process components are loaded on the server side of the configuration.
 - c. Unique Background Process Windowless Client Adaptors, which are loaded by the Background Process Loader as singletons on the server.

6.1 Standalone Configuration

The most basic Viper configuration is standalone, in which all Viper client adapters are loaded in one process on one machine. A standalone Viper configuration may still access remote data sources using a web service or other network protocols to connect to an RDBMS or other data source. In this case though, none of the remote systems will have Viper components on them.

OVERWATCH

6.1.1 Simple Standalone

In the simplest standalone configuration the Viper application and RDBMS are on the same system. In other words, the Viper application accesses a local data store. Figure 6 shows a diagram of this configuration. This configuration is useful for users that must operate disconnected from a network. Data sharing can be achieved by import/export and messaging when connected to a network.

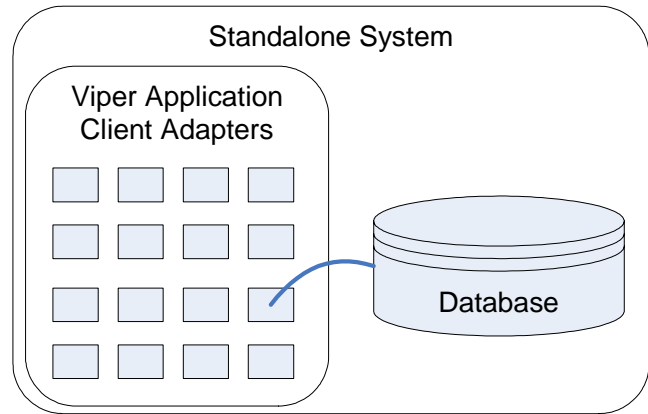


Figure 6 – A conceptual diagram of a Viper application accessing a local data store on a single, standalone system.

6.1.2 Standalone with Remote Network Services

A variation of the standalone configuration accesses data from a remote service, such as an interface to an RDBMS. Figure 5 shows a Viper-based application running on one system while it accesses a database on a remote system. This type of configuration can be achieved by an action as simple as modifying the connection string used by a Viper data adapter to point to a remote RDBMS instead of a local RDBMS. Although simple, this type of configuration does have the disadvantage of there not being an intermediate layer between the Viper applications and the data to ensure that all Viper clients that connect to the same remote database see updates to the data as they occur.

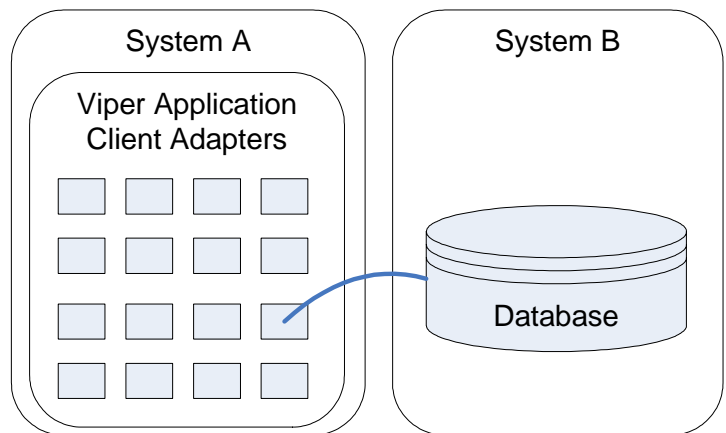


Figure 5 – A conceptual diagram of a Viper application accessing remote data store directly on a remote system.

6.2 Client/Server Configuration

A common Viper configuration is client/server, in which the Collaboration Data Manager (CDM) is used to mediate client/server communications between a Viper Client application and a Viper Server application. Figure 7 shows a diagram of this configuration.

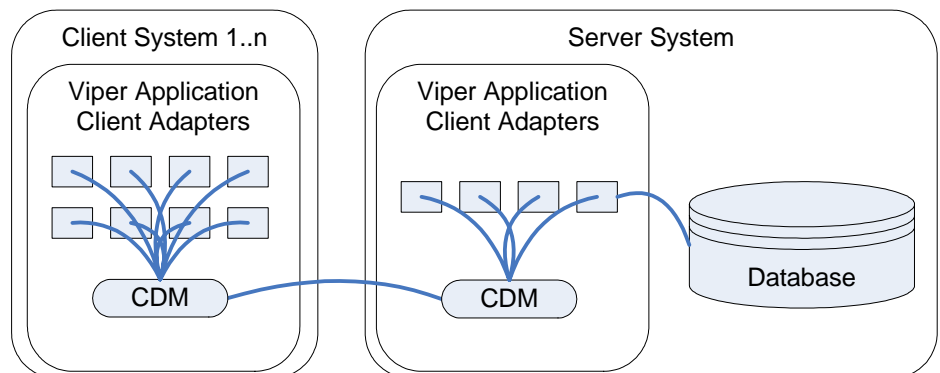


Figure 7 – A conceptual diagram of a Viper client/server configuration.

The advantage to this configuration over the simpler, more traditional client/server approach described in a previous section is that the Viper layer on the server side sees all updates to the data sources and can take action to make

sure that all clients have a consistent view of the data. For example, when data representing an entity is modified, notification of that event is communicated to all connected client systems.

6.3 Viper Client-Server Roaming Client Configuration

The Viper client/server roaming client configuration is a combination of the standalone and client/server configurations, in which the Viper application uses its own local data store when not connected as a client to its server and then switches to using the server's data store when connected as a client. This sort of plug-unplug capability generally requires components for synchronizing the local and server data stores at disconnect and connect time. One approach to accomplishing this behavior with a base Viper configuration is to build Case files before disconnecting and then to synchronize the Case files at reconnection time. For more information on this type of configuration, refer to Section 7.3.

7. Specialized Environments

7.1 High Volume Data Streams

The Viper Framework was developed to support rapid development of C4ISR applications with a priority on supporting collaboration between integrated components hosted in a common UI container as well as collaboration between users. To support this collaboration the Viper Collaboration Framework includes a service bus that manages shared objects and provides publish-subscribe services for sharing events emitted from operations on the shared objects (such as create, read, update, delete operations). At the heart of the collaboration framework is the Collaboration Data Manager (or CDM) which is an in-memory object pool instantiated in each Viper client process which manages shared objects. The Collaboration Framework (CF) provides services that synchronize objects managed by the CDM with persistent stores and, as part of interactive correlation, with objects managed by other CDMs.

Viper's ability to process high volume data streams using a standard Viper client/server deployment architecture (described in Section 6.2) is limited by the overhead associated with supporting collaboration (including marshalling data between CDMs) and the focus on managing non-transitory data. The phrase "high volume data streams" refers to complex event streams (such as those collected by a communications collection device) where the frequency of collected events is high. To support these high volume streams in a multi-user environment, it is recommended that an external Stream Data Service be developed, which persists raw data, provides data filtering based upon client data subscriptions, aggregates data into collections of simpler events, and routes data to clients.

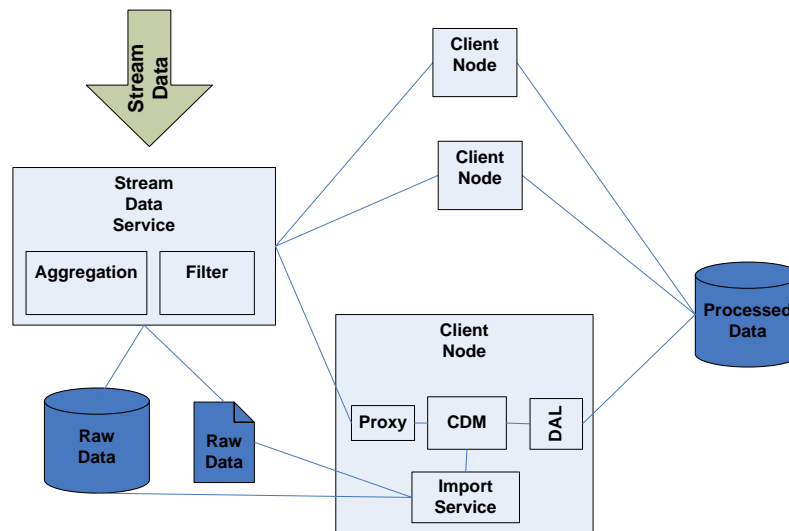


Figure 8: Deployment architecture for multi-user collection processing of high volume data streams.

Figure 8 shows an example architecture for a system performing near-real time high volume collection processing with limited requirements for extensive historical analysis or user collaboration. In this configuration it is anticipated that data is typically segmented based upon some domain specific criteria (such as an AOI and/or frequency range) and clients typically operate on different data segments and won't commonly be performing operations on common data simultaneously. The example architecture includes several components:

- **Stream Data Service:** A specialized or generic stream data processing service interfaces with an external data stream. It persists data to a relational or file based store for archival purposes. This service may filter data based upon system settings (such as filtering SIGNALs data based upon a frequency range) as well as client specific filter settings. Stream Data Services often will perform some initial data aggregation such that only relevant data and metadata extracted from analyzing a series of events is sent to clients. The stream data service routes data to specific client nodes based upon client provided data subscriptions.
- **Client Proxy:** The client proxy is the entry point for pre-processed stream data to enter the Viper Collaboration Framework. The client proxy registers with the stream data service and provides subscription criteria. Typically the client proxy will perform data mediation to map data into Viper collaboration objects that can be operated on by Viper clients.
- **DAL:** Viper data access layer for persisting and retrieving processed data from a common store.
- **Import Service:** Service for importing raw data from persistent stores created by the Stream Data Processing service. Performs data mediation on batch imports.
- **Raw Data:** Persistent stores for raw data populated by Stream Data Services. Provides access to data for non-real time analysis.
- **Processed Data Store:** Persistent store for processed (fused or derived) data.

OVERWATCH

An alternative to the above architecture is shown in Figure 9. Here a server node is introduced that hosts the Viper data access layer and the Import Service. The benefit to this architecture is improved data synchronization across client nodes. This architecture is preferred when operators need to view a common picture of processed data and/or will be operating on common processed data (less segmentation). The single server CDM depicted is conceptual. In reality the Viper server hosts multiple CDMs.

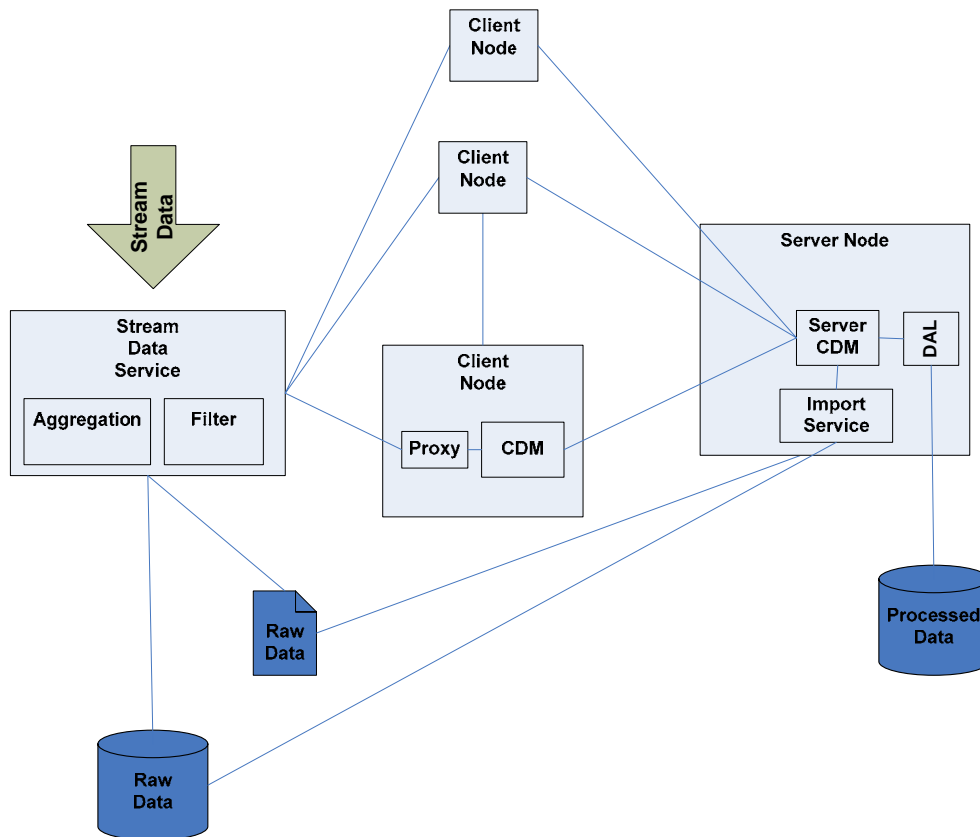


Figure 9: Alternative architecture for high volume data streams.

Note that the architectures described above are based upon the version 1 of the Viper Framework. Work is currently underway to provide reusable infrastructure for integrating high volume data feeds as a Viper collaboration communication channel.

7.2 High Latency

High latency (> 1000 ms) wide area networks are often encountered in distributed C4ISR operations. However, members of C4ISR Communities of Interest often have operational requirements to collaborate on common data and to allow operators to perform the same mission functions regardless of location.

For this operational scenario Overwatch recommends multiple stand-alone or client-server deployments that share data through a custom Data Exchange Service or through the Viper Case File import/export feature. Viper provides significant infrastructure for development of a Data Exchange Service. The function of this service is to monitor data operations on a Viper client/server node, evaluate those operations against mission defined criteria and send the appropriate data asynchronously to a second

OVERWATCH

Viper client/server node. Viper provides infrastructure needed to perform object monitoring, object evaluations using an integrated rule engine, and a reusable data marshalling infrastructure.

In addition Viper provides an “off the shelf” import/export mechanism using the Viper Case Management component. Users can construct Cases made up of Viper managed data and external resources, package them, and send them to a remote node for import.

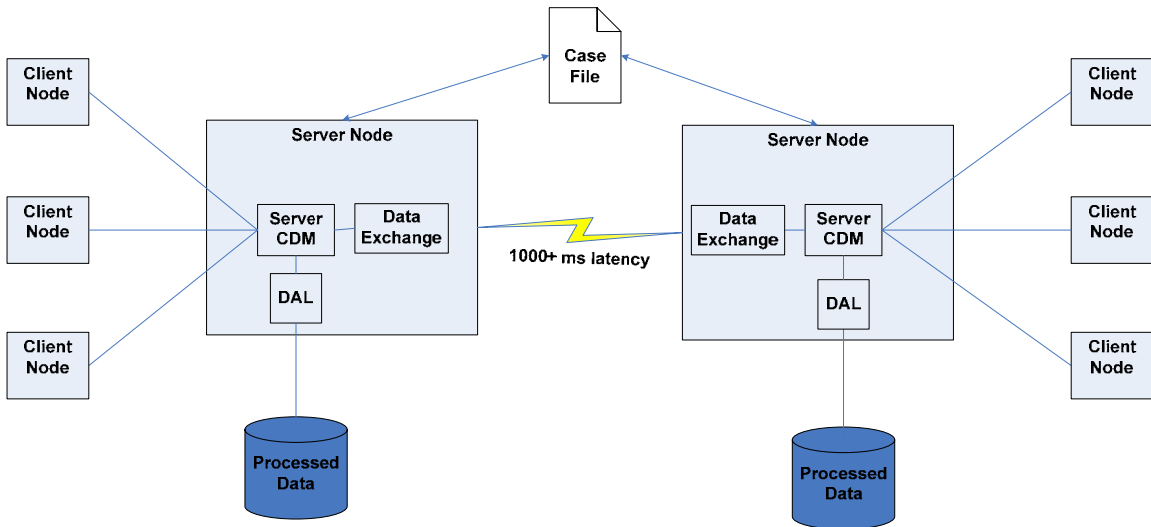


Figure 10: Viper deployment architecture for collaborating clients across high latency wide area networks.

7.3 Intermittent Connectivity

Intelligence On-The-Move operations require systems that can perform while disconnected from a network (i.e., in a stand-alone mode). Viper provides built in support for stand-alone operations on mobile platforms (analyst laptops). However, intelligence operations at the Battalion level often require a mix of connected and disconnected operations. During connected operations users need to have access to common intelligence data stores, a common operational picture and to collaborate with other users on development of derived intelligence.

For these mixed mode operations Viper provides a Case Management feature. While disconnected analysts can enter data to a local data store (see Figure 12) and construct intelligence cases using the Viper Case management capability. These cases contain references to data contained in the local entity store to assure local synchronization of data.

When connected to a network the Viper application running on a mobile platform can be configured to run in client mode and connect to a Viper server to

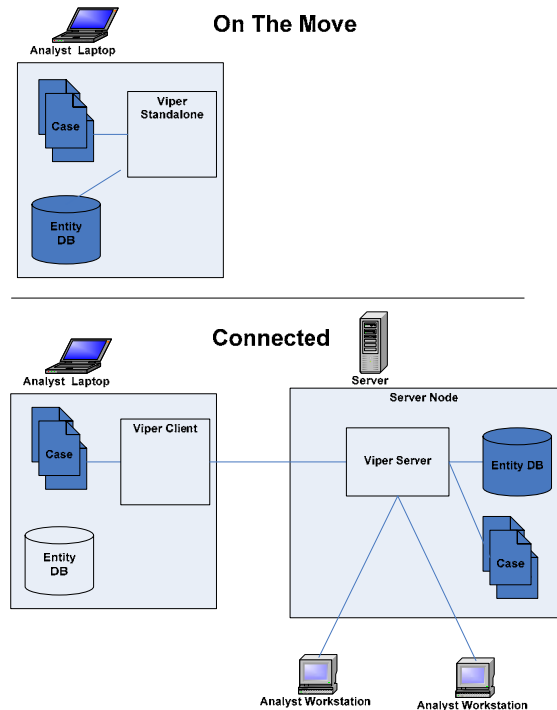


Figure 11: Viper in intermittent connectivity deployments

OVERWATCH

Applications of Viper

analyze data in an enterprise intelligence data store as well as to collaborate with other connected users. Data collected while on-the-move can be shared via multiple mechanisms.

Case files created while on the move can be exported from the analyst laptop and imported into the enterprise intelligence data store. Once this operation is complete the case file will exist on both the local system and on the enterprise system and data contained in the case will exist in both the local analysts store and the enterprise store.

Case files created while on the move can be migrated to the server. Once this operation is complete the case file will exist only on the enterprise system, however data contained in the case will exist in both the local analysts store and the enterprise store.

In addition the analysts can export data from the enterprise intelligence store to a local case for use while on-the-move.

7.4 Thin Clients

The term “thin clients” typically refers to applications in which the total application “presence” on user nodes is minimized. Presence refers to several aspects of software deployment and execution:

- Resource consumption
 - Storage space
 - Memory
 - CPU
 - Bandwidth
- Configuration, Install and Security
 - Requirements for local installation and configuration.
 - Permissions and system access needed to install and run software.
 - Ability to update software remotely/automatically.
 - Dependence on open network ports on clients and non-standard protocols.
- Pre-Requisites
 - Dependencies on non-standard external software installed locally – particularly software that requires additional license fees, complex configuration and install and/or OS extensions.
- Persistence/Roaming
 - Persistence of user, session, or application data to the local system.
 - Ability to access user, session, or application data from multiple systems.

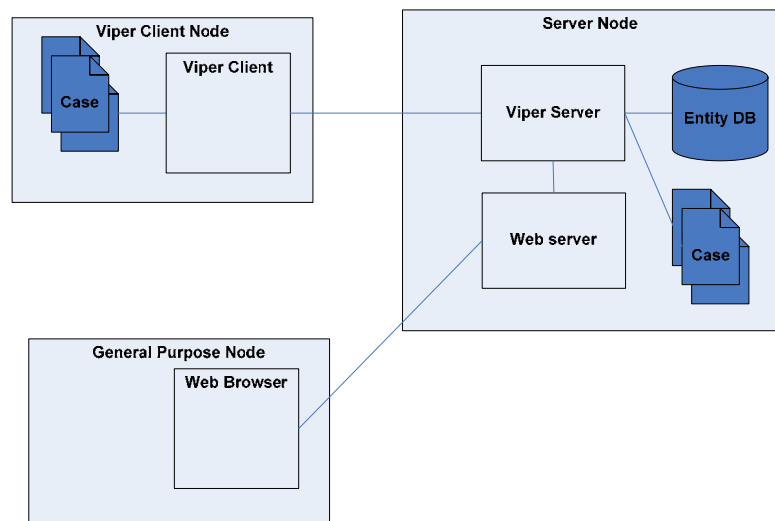


Figure 12: Viper Thin Client Deployment

In many cases the term “thin client” is used interchangeably with “web client” and refers to a client application that can run within a web browser. The Viper Framework is currently being extended to

support a “thin client” interface. . Thin client support is being delivered in stages based upon OWS customer requests.

7.4.1 Near Term

A web based Entity Browser has been developed that allows users on general purpose or disadvantaged network nodes to access Viper data. Disadvantaged network nodes are those that cannot support a Viper application install due to platform incompatibilities or resource limitations. This entity browser provides basic grid and detailed views of entity data managed on a Viper Server (as shown in Figure 13) and provides full text searching of Viper data sources. In Viper 1.4 users are able to perform operations on all Viper entity data via this interface. However, Viper collaboration is not supported via the web Entity Browser and common Viper analytical tools such as an integrated map display are not available. No client side SDK is currently available, however Web Service interfaces are provided for development of custom web clients.

7.4.2 Web Framework

Overwatch will develop a web-based Viper UI framework for developing interactive, collaborative collection and analysis applications. The web-based UI framework will provide features similar to the existing Viper UI Framework:

- Client collaboration (within a single web page) – Event driven collaborative interactions for drag and drop, copy-paste, and coordinated selection.
- Multiple Document Interface – Multiple UI components presented to the user simultaneously in separate containers within a single window. Support for resizable containers with user defined layout.
- Modes/Preferences – User defined display preferences and modes. Modes define the set of tools loaded into the framework, their layout, and collaborative behaviors.

Web based Viper applications can share data, views, and services with smart client Viper applications running on the same network.

The web-based Viper UI framework will include web application technologies for producing dynamic, collaborative user interfaces – including AJAX, DHTML/XAML. Where possible the framework will make use of SharePoint Portal services. Microsoft SharePoint Portal services include the Web Part Toolkit. The Web Part Toolkit is a portlet development kit that leverages the Web Services for Remote Portlet (WSRP) specification. WSRP defines a protocol for developing presentation-oriented web services that are aggregated by a web portal application. These presentation oriented-web services generate markup fragments that are wired together on the client to create a composite application. The Web Part Toolkit provides infrastructure for rapid development of portlets – including mechanisms for defining client-side messaging between portlets. Web Parts can support user customization of the interface through drag-and-drop addition of controls.

Major architectural elements of the web-framework include:

- Web browser client, AJAX-based framework that publishes and subscribes to objects, events and interfaces.
- Provides a plug-in framework for web-based clients/portlets.
- Support for multiple windows (within single application browser page), split panes, drag and drop between windows. No Frames.

- Where appropriate, supports rich interactivity such as panning, scrolling, zooming and the drawing of data entities and arbitrary polygons.
- Utilize server-side .NET Web Controls to allow a developer to quickly add components to the web application that will emit browser-specific code.
- Client-side module container that manages client-side objects and subscriptions (a.k.a. Web Client CDM).
- User interface events collaborated with other clients (including non-web clients).

7.5 Mixed Platform

The Viper framework utilizes Microsoft .NET components and is intended to be deployed on a Windows platform. In most enterprise computing environments there is a mixture of operating systems (e.g. Windows and Linux) and application platforms (.NET and Java). There are numerous ways that Viper can be mixed with non-Windows or non-.NET applications and services. The most common integration requirement involves integration using web services and integration with Java. While Viper client adapters written in .NET are generally the preferred approach for interoperating with Viper, it is possible to access Viper services through other mechanisms such as web services or JNI. In such cases, third party tools may have to be brought to bear, although starting with a Viper client adapter is the recommended approach.

7.5.1 Integration via Web Services

The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web. The W3C defines a Web Service quite broadly as "a software system designed to support interoperable Machine to Machine interaction over a network." The common technical usage of this term is more narrowly scoped to "a software system designed to support client to server interactions over a network using XML in the SOAP format over the HTTP protocol where the server's operations are described in a machine readable way using a Web Services Description Language (WSDL) document." HTTP, SOAP and WSDL are all W3C standards. There are many tools available that allow both .NET and Java applications to easily interoperate with Web Services using these standards. The availability of such standards is what makes this the preferred mechanism for interoperating between Viper and other systems.

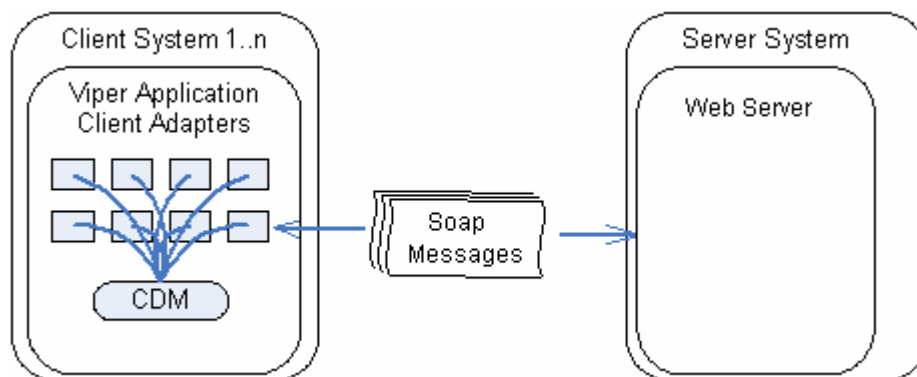


Figure 13 – SOAP provides the bridge between a Viper client and an external web service.

When integrating with web services, the common approach is to implement a Viper Client adapter that translates between Viper's view of the data and the Web Service's view of the data.

7.5.2 Direct Integration of Java

Overwatch has another white paper that thoroughly covers this subject: “A Discussion of Viper/Java Interoperability”. Here, the focus is on the main recommendations from this paper. There are two categories of direct .NET to Java integration: .NET calls Java and Java calls .NET.

7.5.2.1 .NET Calls Java

There are a number of ways that a .NET application can directly invoke code written in Java. Some of these require adaptation on the .NET side, some on the Java side, some on both sides. The preferred option allows for .NET to call Java directly with no changes required for the Java code using the IKVM.NET open source project.

IKVM.NET is an implementation of Java for Mono and the Microsoft .NET Framework. It includes:

- A Java Virtual Machine implemented in .NET
- A .NET implementation of the Java class libraries
- Tools that enable Java and .NET interoperability

IKVM.NET is the ideal solution when:

- Complete computational integration is required
- Minimal, or no, GUI integration is required
- The developer is .NET oriented
- Performance is important

The Java code is actually executed in the .NET engine with this approach. If access to a real Java VM is needed then this approach will not work. Also this approach can not be used to run UIs developed in Java.

7.5.2.2 Java Calls .NET

The preferred approach to enable Java to call .NET is to create .NET components on the Viper side having Microsoft Component Object Model (COM) interfaces. Microsoft’s Component Object Model (COM) is a platform for software components introduced by Microsoft in 1993. It enables interprocess communication and dynamic object creation in any programming language that supports the technology.

It takes only some simple annotation when defining a .NET class to also make that .NET class available as a COM component. Following the creation of the .NET library containing COM components, the COM components can be made available to the system using the Microsoft regasm.exe tool.

It has always been possible to call COM components by using Java’s JNI facility. Due to the ubiquity of COM components on the Windows platform a number of tools and libraries have evolved to make the interoperation between Java and COM easier: Jacob, COM4J or Tlb2Java are some of these tools.

7.6 Protection Level 3 (PL3)

Protection Level 3 (PL3) applies to information systems that maintain, process, or transmit intelligence information. Intelligence information systems are accredited at a Protection Level from 1 to 5 depending on the usage of the system. A system must be accredited at PL3 when at least one user lacks at least one required formal approval for access to all the data on the system. This means that all users of the system

have all required clearances, but at least one user lacks formal access approval for some of the information on the system.

An important point to note is that a complete software system is accredited at PL3, not the Viper infrastructure itself. Viper is one of many system elements that must be taken into account when PL3 is a system requirement. The first PL3 software system based on Viper is scheduled to be certified at PL3 in June of 2008.

For a system to be accredited at PL3 it must meet security requirements in several areas including access control, account management, auditing, authentication, data transmission, recovery, resource control, session control and data storage. Viper allows a system to meet these requirements. In certain cases, such as peer-to-peer network configurations, additional architectural elements may be needed to meet both operational and PL3 requirements. For example, Viper clients are not allowed to have open network ports in a PL3 environment. If Viper clients must connect to each other, as in peer-to-peer connections, such connections must be proxied via a Viper server.

The PL3 requirement that most affects current Viper implementations in the PL3 environment is the PL3 requirement that “information is properly marked and labeled”. Currently, Viper applies security markings to entities at the entity level only – their individual properties can not currently have separate markings. This is a data source specific issue and Viper’s current implementation is sufficient for some, but not all data sources. Support for broader marking capabilities is currently being added to Viper and will be present before June 2008.

7.7 MLS

Multi-Level Security (MLS) is an issue when a software system must process information at different security levels, permit simultaneous access by users with different security clearances and different needs-to-know, and prevent users from gaining unauthorized access to data. MLS support requires that data entities, parts of data entities and parts of prose be able to be marked with disparate security markings. Documents and entities must be filtered so that the user viewing data sees only data that they are cleared to see despite the fact that the full document or entity contains data that the user is not cleared to see. Full MLS support is a requirement for PL4 environments.

Viper currently supports marking entities using a pluggable security infrastructure. Viper is being extended to support Protection Level 3, and by June 2008 Viper’s pluggable security infrastructure and UI components will support per-field markup and paragraph markings in text documents which will lay the ground work for full MLS support. A product fully supporting MLS requirements could be implemented on Viper by providing a mechanism for tracking the clearance level of the logged in user and by making sure that data drivers properly filter data to that which the current user is allowed to see. If this product wanted to use Viper’s collaboration features, the collaboration framework would require modifications to assure that the collaborators had the same clearance levels.

7.8 Safety Critical

A safety-critical system is a system whose failure or malfunction may result in:

- death or serious injury to people, or
- loss or severe damage to equipment or

- environmental harm.

Software that is developed for military safety-critical systems is developed to the strictest standards under a highly detailed review process.

While Viper itself will never be developed under the rigor required for Safety Critical Software, it can be used as a front-end to safety critical systems and using system virtualization it can even be run as part of a Safety Critical System.

One possible configuration would be to run Viper in a MS-Windows ‘padded cell’ on top of the Integrity RTOS while safety critical code executes in its own partition.

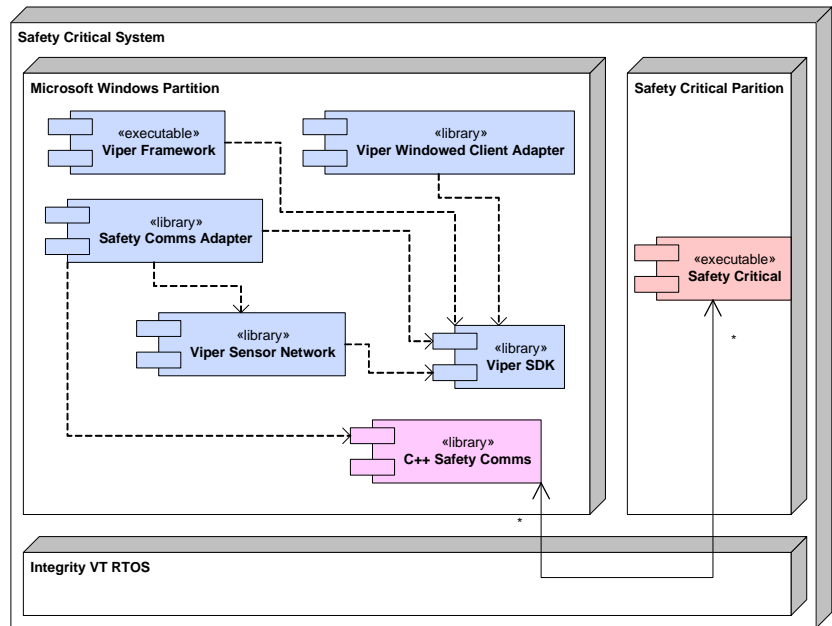


Figure 14 Viper on top of Integrity RTOS

The Green Hills Software Integrity RTOS provides an operating environment where safety critical code and non-safety critical code are separated into isolated partitions. Each partition can have its own virtual machine which allows a full installation of a Microsoft operating system in one partition that is guaranteed to not be able to interfere with the operations in the safety critical partition. The Integrity RTOS provides a pipe-like communication mechanism between partitions. Figure 15 shows a Viper client adapter that uses a C++ library to perform communications between the safety and non-safety critical partitions.

8. Conclusion

Viper is a flexible software framework that can be used in a variety of scenarios to support the construction of C4ISR applications. Viper implements a product line architecture and serves as the foundation of the Overwatch Intelligence Center (OIC) software product line (SPL). Viper can be used as the foundation architecture for a product line, be used to create composite applications and be used to create smart applications. Viper-based systems can be deployed in several different network configurations including standalone, client/server and a combination of both. Viper can also be used in specialized environments such as those with high volume data streams, high latency, intermittent connectivity, thin-client requirements, mixed technology platforms, PL3 requirements, multi-level security requirements and safety critical requirements.

9. References

[1] P. Clements and L. M. Northrop, Software Product Lines: Practices and Patterns. Addison-Wesley, 2001.
 [2] Product Line Approach to Weapon Systems Acquisition, Col. J. Michael Hanratty, Ph.D., Open Systems Joint Task Force, Office of Undersecretary of Defense, Acquisition and Technology, Dr. James H. Dixon, WALCOFF Technologies, Charles K. Banning, WALCOFF Technologies, CrossTalk, The Journal of Defense Software Engineering. <http://www.stsc.hill.af.mil/crossTalk/2000/11/hanratty.html>.

