15-499A Concurrent Engineering

Carnegie Mellon University

School of Computer Science

Bernd Bruegge

DIAMOND Problem Statement

16 January 1996

Version 1.0

Abstract

The goal of the DIAMOND (Distributed Intelligent Assistant for Maintenance ON Demand) system is to improve and optimize the flow of information between engineers performing maintenance on people mover systems. A first prototype of the system was built in 15-413 Software Engineering in the Fall 1995. The goal of 15-499A is to continue the development of the system, focusing on the functionality and the system design described in this document and deliver it on May 2, 1996.

Revision History

1.0 16 January 1996. Bernd Bruegge, CMU.
Initial Release.

Table of Contents

1. Introduction

The growth of airline passenger traffic in the last decade has created a demand for larger airports. Unfortunately, large airports create two problems. First, travelers must walk long distances to get to the departure gate and baggage handling areas. Second, a considerable amount of time is spent by the airplanes moving on the ground between passenger gates and runways wasting large amounts of fuel and money.

A solution to both of these problems has been the redesign of airports coupled with the introduction of Automated Guided Transit Systems (AGTS)—people movers. People movers are automated trains that move passengers efficiently over long distances. People movers have two advantages. First, they make it easier for passengers to move across distances in airports, and second—when placed underground—they eliminate traffic from the surface allowing planes to move more optimally while navigating to and from runways. As a result, airport redesign and people movers significantly reduce the fuel requirements for planes.

By placing people movers between gates and arrival/departure areas, the people movers become a crucial part of airport operations. In a busy airport, any problem that causes down time in a people mover has the potential of disturbing thousands of passengers. Therefore, a major operating requirement for people movers is extremely high availability.

The goal of the DIAMOND (Distributed Intelligent Assistant for Maintenance ON Demand) system is to improve and optimize the flow of information between engineers performing maintenance on people mover systems. A first prototype of the system was build in 15-413 Software Engineering in the Fall 1995. The goal of 15-499A is to continue the development of the system, focusing on the functionality and the system design described in the following sections.

DIAMOND is part of a larger system development effort called AMEFS (ATS Maintenance Engineering Feedback System) The overall AMEFS is expected to provide easy access to an integrated information space so that technicians, engineers, managers and trainers can author or obtain needed information anytime and anyplace their work demands it. The scope of AMEFS is defined in terms of six scenarios defined in the "AMEFS Requirements Document"[1].

2. System Overview

Figure 1 shows the boundaries of the DIAMOND system. DIAMOND consists of an Information Space, a set of Agents operating on the Information Space

[1] The AMEFS Requirements Document is available on the Lotus Notes Database "15-499 Documents", Category AMEFS.

and a set of Presentors. The **Information Space** interacts with a set of Authoring Tools via **Converters**. The **Authoring Tools** themselves are not part of the system.

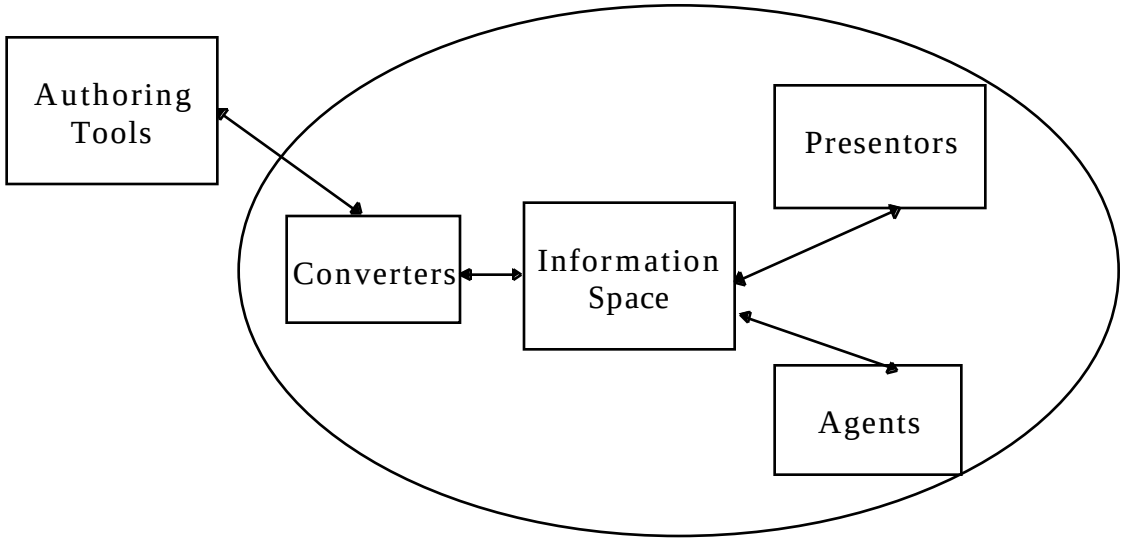


Figure 1 System Boundary

The Information Space is a collection of documents called HyperDocuments. The Information Space allows the management of HyperDocuments for a single user or a group of users. Creation and editing of some of these documents is done with Authoring Tools, other documents are created within the Information Space. The **Presenters** provide mechanisms to view HyperDocuments in the Information Space. Different types of users, such as engineers at desktop workstations and technicians with mobile computers such as laptops, PDAs or and wearable computers will of necessity have different views of the documents through appropriate interfaces. Agents operate on the Information Space. Examples of **Agents** are diagnostics systems that look for anomalous behavior or information search engines that look for the occurrence of certain events.

The overall view of the system is shown in Figure 2. HyperDocuments are stored persistently in the DIAMOND database, which provides a uniform and consistent interface to a federation of databases such as EDMS drawings, work orders, electronic forms, technical documentation and training material. Users will access HyperDocuments for a number of applications including preventive maintenance, predictive maintenance, corrective maintenance and training.

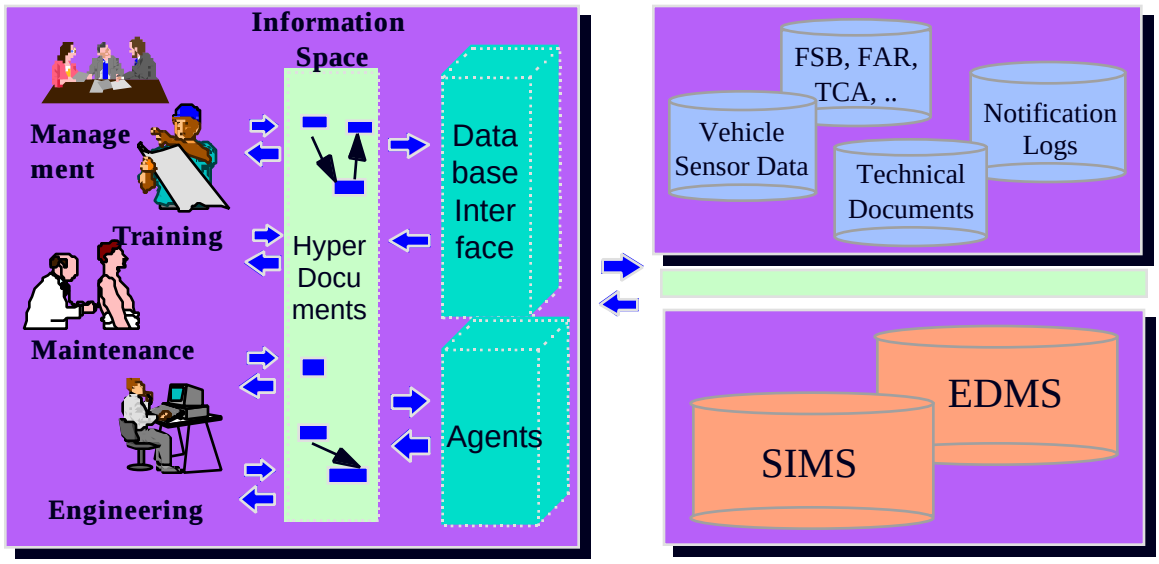


Figure 2 System Overview

The document is organized as follows. Section 3 describes functionality offered by these applications as part of work package 2. Section 4 describes the major abstractions of the system, in particular the Information Space (Section 4.1) and the HyperDocument (Section 4.2).

A major design goal of the system is provide support for a variety of users in different environments ranging from engineers at headquarters, field site engineers at a field site to maintenance technicians working on a people mover. The user interface of the system is described in Section 5.

The architecture is described in Section 6 and is based on a suite of robust, interchangeable modules which can be easily ported to different systems and platforms. Section 7 describes our decisions with respect to the hardware platforms and software building blocks used in the development and deployment of the system. Section 8 describes the authentication and security provisions. The database management system is described in Section 9. Section 10 describes the physical connectivity of the system covering wired and wireless networks. Section 11 contains a description of the boundary conditions of the system, in particular system initialization and system behavior during failure and termination conditions. Section **Error! Reference source not found.** concludes with a summary of main features of the system. Appendix 1 contains a more detailed description of the use cases listed in Section 3.

3. Functionality

In the following we define the functionality of DIAMOND II as a set of use cases .

3.1 Information Space Use Cases

The information space use cases provide a common set of operations when entering and exiting the system as well as when manipulating documents. The information space supports a variety of documents such as vehicle data, images, structured text described in detail in Section 4.2.

- **Login**: The user authenticates by issuing a user name and password and enters the information space.
- **Add Document:** This moves an external document (authored by another Tool) into the information space.
- **Remove Document**: This removes a document from the Information Space
- **Query**: The user issues a query returning a set of documents. which are available in the Information Space.
- **View Document:** Display a document.
- **Edit Document :** Change a document.
- **Create Link to Document:** Create a link to another document.
- **Delete Link to Document:** Delete link to other document
- **Navigate**: Follow a link in one document pointing to another document.
- **Share Document:** This use cases allows for collaboration of different users over documents. This enables one user to share a document and its annotations with another user at a remote location. The user invokes a dialogue to determine with whom to collaborate. The collaboration information and the selected document such as an EDMS image are sent to a collaborators machine, which is started and displays the selected document.
- **Zoom Document:** Zoom into a (shared) document. (The zooming is done under software control, not under user control).
- **Logout**: The user exits the information space. Documents currently in the information space are kept for the next session.

An important class of documents is the structured text which is dealt with by a separate set of use cases for forms.

- **Initiate Form:** The user chooses a blank form template (e.g. FAR), fills it out and sends it on to the appropriate employees
- **Process Form:** The user reads an incoming and responds to it.
- **Query Form Status:** The user checks the progress that has been made on an previously initiated form.
- **Search Forms:** The user searches and retrieves forms from a database.
- **Traverse Form Link:** The uses moves from one form to another by traversing an existing link on a form.

- **Navigate Report:** This sub-usecase in which a user examines a generated report is part of several of the usecases above.

3.2 Diagnostics Use Cases

These use cases implement the predictive Maintenance scenario in the DIAMOND requirements document. The data will come from sensors in the doors and propulsion system.

- **Transmit sensor data**: A People Mover with data sensors transmits data generated by these sensors to the Notification Server. Examples of sensor data include the opening or closing of doors, a door failure, or the speed of the People Mover.
- **Receive sensor data in real time**: Sensor data is received in real time by headquarters (either West Mifflin or a simulated headquarters in the Vehicle Systems Lab at CMU) where it is viewed by an engineer.
- **Store sensor data:** Sensor data is stored as persistent data in a vehicle data database.
- **Retrieve sensor data from the same train at different times :** The diagnostics engineer searches the vehicle database for sensor values for a given train and airport for two time ranges specified by start and stop times.
- **Retrieve sensor data from different trains:** The diagnostics engineer searches the vehicle database for sensor values for a given airport for two different trains and two time ranges specified by start and stop times.
- **Retrieve sensor data from different trains in different airports:** The diagnostics engineer searches the vehicle database for sensor values for a given airport for two different trains and two time ranges specified by start and stop times.
- **View sensor data offline:** A graph of sensor data is produced and viewed, using one of the view options such as time line, histogram or pie chart.
- **View sensor data online:** View sensor data. View is updated in real time as new sensor data is generated.
- **Detect anomaly in sensor data:** The sensor data from different times, different trains and different airports are analyzed. An abnormal door closure time, which is normally less then 5 seconds, is detected by the diagnostics system and an event is sent to the Notification Server. An engineer who has subscribed to the occurrence of anomalous door closure times is notified by the diagnostics system of the occurrence of such an event.

3.3 Preventive Maintenance Use Cases

These use cases are necessary to implement the Routine Maintenance scenario described in the AMEFS requirements document.

- **Retrieve a Manual**

· **Link Text to Related to Documents .** The maintenance technician creates link in a paragraph in the manual to a FAR or an EDMS image.

Search for Text in Manual. The maintenance technician searches for the occurrence of a specific string in the manual.

3.4 Corrective Maintenance Use Cases

This use cases are necessary to implement the fault isolation part of the Routine Maintenance/Fault Isolation Scenario described in the AEMFS requirements document.

· **Retrieve an EDMS drawing**

3.5 Engineering Feedback Use Cases

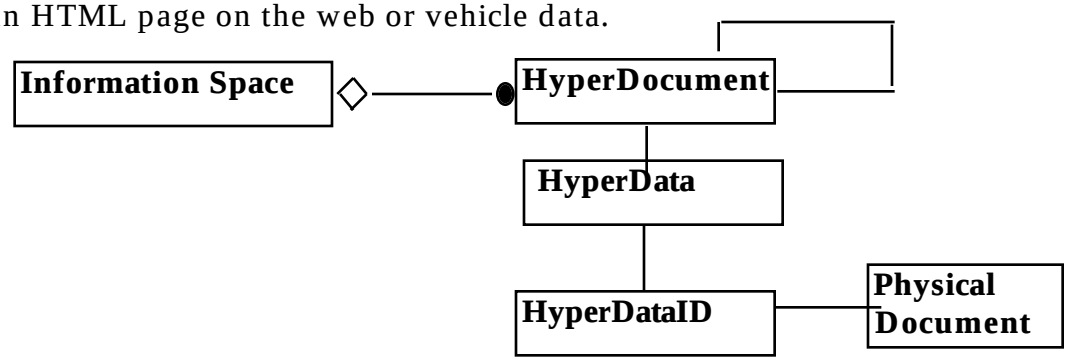
The use cases to support feedback to engineering focus on providing data from predictive maintenance and notify engineers and field site managers of new information.

- **Initiate Information FAR:** A field site manager has information that needs to be shared with other sites. The field site manager creates and sends an information only FAR to headquarters containing this information. information
- **Initiate Critical FAR:** A problem at the field site requires immediate attention. A field site employee describes the situation by creating a critical FAR and sends it to headquarters.
- **Initiate Quality FAR:** From experience at the filed site, an employee develops an idea for product improvment. The field site employee creates and sends a quality (enhancement) FAR to headquarters for later use by design engineers.
- **Query Information Space:** A design engineer is engaged in designing a subsystem for a new product. The engineer searches the information space to retreive information from the field relevant to that subsystem. This may include quality FARs relating to that subsystem, the collection of work orders automatically generated by the diagnostic system (to understand the pattern of breakdowns for certain components), and any other relevent information sources in the information space.(relevant FSBs, incident reports, etc.)

• **Automatically Generate Work Order:** The diagnostic subsystem determines that a maintenance procedure is needed for the continued running of a vehicle. It automatically creates a work order for the field site employees to handle.

4. Object Model

This section describes the important abstractions provided by the system. The central component of the system is the Information Space which is a collection of zero or more HyperDocuments[2].
Each Hyperdocument is more or less a set of links (linking it to other Hyperdocuments pointing to an abstraction which we call HyperData. Associated with Hyperdata are its Representation and a pointer to the actual data itself which is modeled as a class called HyperDataID. HyperDataID classes are references to the actual actual data available for manipulation by DIAMOND. Examples are EDMS images, work orders, training documents, an HTML page on the web or vehicle data.



Users can populate the Information Space with HyperDocuments, manipulate them, and remove them from the information space. HyperDocuments can be linked to other HyperDocuments and shared with other users. The system is document-centric: everything in the Information Space, be it an EDMS drawing, a work order, a field service bulletin, or a training manual, is a HyperDocument.

We start our discussion with the Information Space in section 4.1. The HyperDocument is described in section 4.2. Different types of data can be attached to HyperDocuments described in Section 4.3.

4.1 Information Space

The Information Space class shown in Figure 3 The Information Space provides several attributes and functions for the DIAMOND user, in particular:

---

[2] The notation we use for describing classes and object models is based on the OMT notation. Each class is represented as a rectangle with up to 3 boxes separated by lines: The first box specifies the name of the class in boldface. The second box contains the set of all attributes. Each attribute consists of a name and its type, separated by a colon (:). In the cases where the type is left out, it will be determined during the object design phase. The third box contains all the operations defined for this object. If the input/output parameters and return types for the operation have not yet been defined the name of the operation is postfixed with an empy parameter list "()". In some cases, where we are only interested in the relationship of objects to each other, the attributes and the operations are left out and the class is shown as a single box containing only its name.

- Authorization of Users
- Queries on Hyper Documents
- Addition and Removal of Hyper Documents

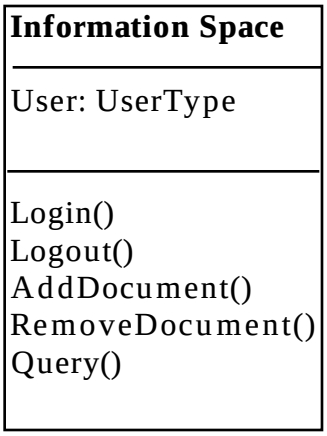


Figure 3 The Information Space Class

Authorized users are allowed to enter the information space and perform operations such as submission of newly created data by Authoring Tools into the Information Space. **AddDocument()** involves conversion of the newly authored data into a HyperDocument, assignment of a Security Level, and connection of the HyperDocument to the data itself (data is not part of the HyperDocument). Users can also issue queries.

Query() returns a set of HyperDocuments which can then be manipulated using operations for viewing, annotation or sharing of the document with collaborating users.

4.2 The HyperDocument Class

A HyperDocument consists of several attributes and operations as shown in Figure 4.

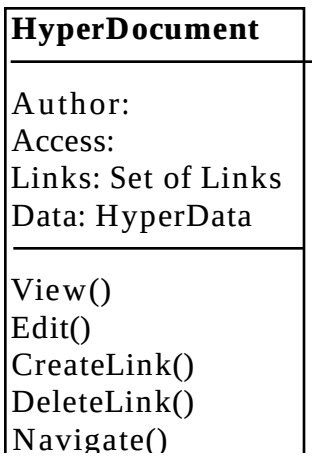


Figure 4 HyperDocument Class

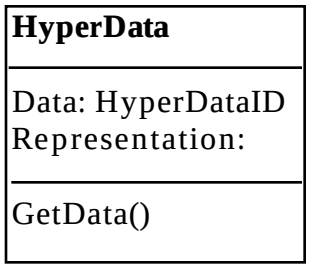
The **Author** attribute is a reference to the user (or process in the case of a Legacy System conversion) who created the document. The **Access** attribute defines the security level for the document. An entity in the system (a user or group of users) may have read or write access, both, or neither.

The **Links** attribute is a set of links to other HyperDocuments. The **Data** attribute is a reference to where the actual data associated with this HyperDocument is stored. Because the system has to support many different data types, some of these created by external authoring tools, **Data** is modeled by another object called **HyperData** which is described in Section 4.3.

The **View()** operation is used to obtain a system-independent model of the data within the object, which may then be rendered on a particular platform by the user interface. This object uses HyperData's various methods to accomplish this. The **Edit()** operation works in a similar fashion, launching the appropriate application editor based on the HyperData's type[3]. The **CreateLink()** operation is used to create links between a regular HyperDocument type and another HyperDocument. **DeleteLink()** removes a user created link. **Navigate()** follows a given link. **Share()** makes the Hyperdocument viewable by another user and **Zoom()** provides the capability to move into more detail of the document given a certain zoom level.

4.3 HyperData

As mentioned above, the **Data** attribute of HyperDocument is of type **HyperData**, which is shown in Figure 5.



[3] We have do not have all the attributes we need to be able to handle issues like multiple editors on the same HyperDocument, etc.. perhaps an 'Editable' flag, or 'Multi-Editable' flag is necessary.

Figure 5 HyperData Class

HyperData provides the operation **GetData()** which looks at the Type field, to determine the type of representation. Different representations will be supported such as SGML, HTML, MPEG, TIFF, EDMS and "Unknown Type". For instance, if Netscape or another WWW browser is being used, the HyperDocument object calls the **GetDoc()** operation which returns the document in HTML format and then asks Netscape to display it. HyperData of Representation SGML allow more sophisticated viewing methods with SGML browsers or editors. A work order may be returned in SGML (Standard Generalized Markup Language) format allowing more powerful view capabilities with SGML browsers. Data types from existing applications, such as a Microsoft Word document, are of type "Unknown". They may be viewed by launching the appropriate application to display the document. All data types supported by the DIAMOND system are modeled as subclasses of **HyperData**. Initially the system will support the types shown in Figure 6.

- **Vehicle Data:** An array of Sensor Data. Sensor Data contains a vehicle identification id (airport, train, car, sensor id), a time stamp and a sensor value.
- **Structured Text:** Examples of structured text are Work Orders, End of Shift Reports, Field Service Bulletins (FSB), Temporary Change Notification (TCA), and Engineering Change Authorizations (ECN). Structured Text will be in SGML format. An SGML DTD (Document Type Definition) must be defined for each of the supported document types. A consequence of this representation is that the particular fields of a structure text type - the date of a Work Order, for example - may be accessed individually. As a result, many documents (Work Orders, FSB's, etc.) can be manipulated consistently.
- **Image:** This data type provides supports for "legacy data" which may not be easily integrated into the system (for example because of proprietary data formats). An Image is either a raster image or a bitmap. An example of a bitmap is a scanned EDMS drawing, an example of a raster image is the AutoCAD version of the EDMS drawing.

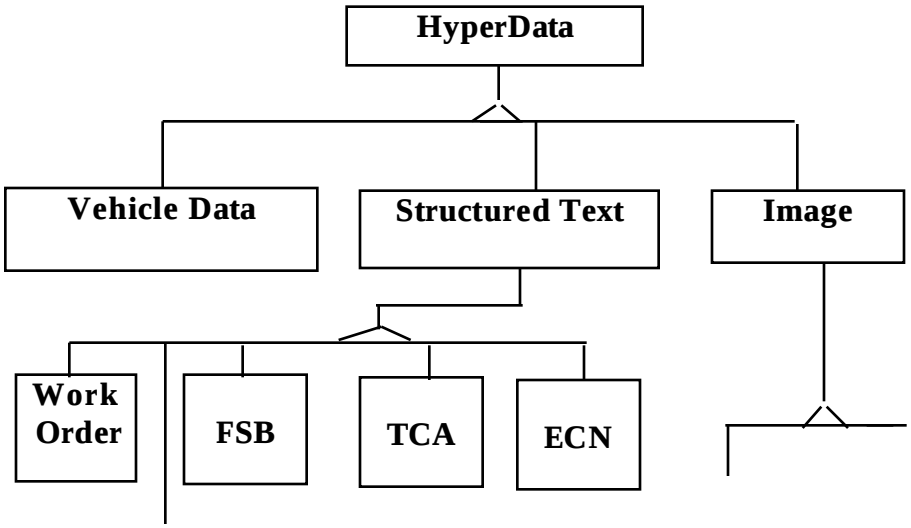


Figure 6 Types of HyperData (Type Hierarchy)

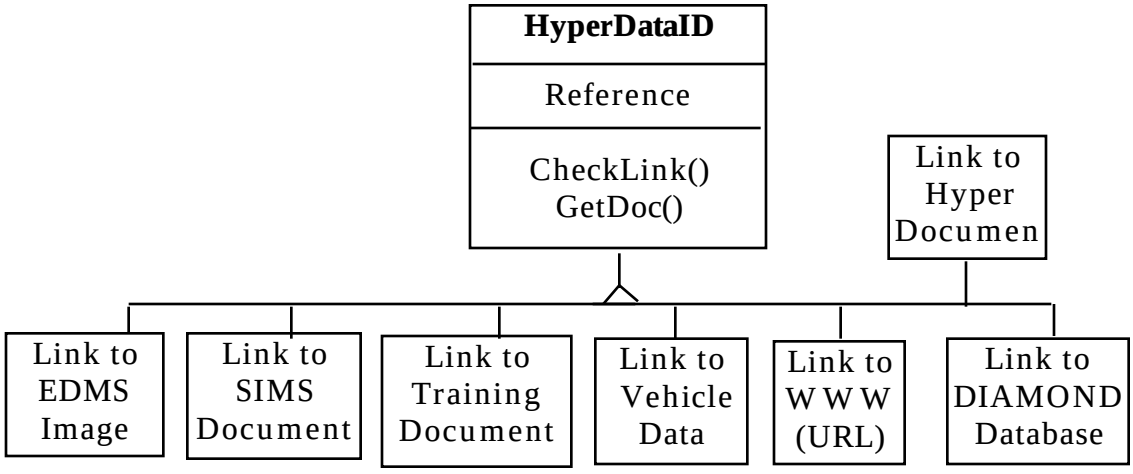
New data types such as **Video** can be added by inheritance, which involves the implementation of the HyperDocument operations for the new type. If any of the HyperDocument operations do not make sense for the new type, the specific operation has to be masked out using the technique of method overriding or method redefinition.

4.4 HyperDataID

Note that neither the **HyperData** object nor the **HyperDocument** object contain the *actual* data; they contain only the "hyper" information describing the connectivity of the data with other data in the information space and outside.
The **HyperDataID** class contains the reference to where the actual data associated with this document is stored. HyperDataID can be one of the following:

- a link to an EDMS image in the vault,
- a link to a work order in SIMS
- a link to a training document
- a URL pointer into the world wide web
- a link to data stored in the system database (Vehicle Data, Notification Log, ...)

Several operations are defined on **HyperDataIDs**. **CheckLink()** allow a user to check the validity of a link. **GetDoc()** retrieves the physical document associated with the HyperDataID.



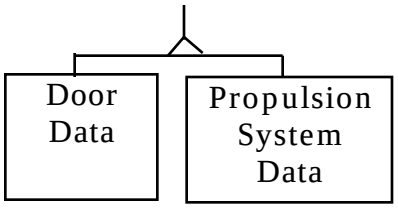


Figure 7 HyperDataID: The Link to Physical Documents

Links are allowed to change at runtime. The change can be the result of different security level assigned to the document (making a document available for a different audience) or a change due to configuration management issues (relocating the location of a physical document).

5. User Interface

The user interface for the system has the following features:

- Netscape 2.0 provides the primary user interface.
- All possible standard menus and buttons of Netscape are hidden so that users are not confused with Windows concepts.
- There are two main windows: one is a data window and the other is an information window. The information window contains instructions and lists that the operator can choose from to continue a dialogue. These are not windows in the Windows sense but in the Netscape sense.
- There are a small number of possible initial screens that correspond to the primary tasks that users perform with the system. Each user has one of these initial screens brought up when they log in.

In general, the principles that govern the user interface design are simplicity and limited windows. The trade off is that is it might take several actions to perform a task and the benefit is that at any point, the possible actions are clear.

An open issue is that user interface that is used for white board collaboration. This depends on the particular system chosen for the white board. Ideally the system would have an API so that collaboration could be accomplished within the context of Netscape and a user would not need to perform any special actions other than pushing a "collaborate" button to do the collaboration.

6. System Architecture

The system architecture is modeled as a collection of three sets of collaborating processes as shown in Figure 8. The first set contains

DIAMOND-specific applications called workbenches. Each workbench itself consists of two components, the user interface and the application. Different workbenches will have different look and feel depending on the platform and use. In particular, a workbench can offer a different user interface depending on whether it is running on a desktop or mobile computer such as a laptop or PDA. The access to the various applications is through a general user interface based on Netscape as described in Section 5. The user interface design distinguishes between commercially available applications with existing user interfaces such as Spicer's Animation for the viewing of EDMS images as well as applications, whose user interfaces are written in the context of the DIAMOND project. Viewers and editors within Netscape will depend on the data type. Access to viewers and editors is provided via Netscape applets.

The second set of processes provides the capabilities for information space management, visualization, query management and collaboration. In some sense these are applications, but they are independent from the DIAMOND application domain. The Information Space Manager manages the move of documents between the authoring environments and the information space. Visualization provides the capability for visualization of data in different formats. The Query Manager decomposes a query into its database constituents and sends these to the corresponding databases managed by the Data Server.

The third set of processes consists of servers providing access to persistent storage (Data Server), security services for authentication (Security Server), mobile transmission (Communications) and notification allowing applications and servers to subscribe to interesting events and be notified of their occurrence.

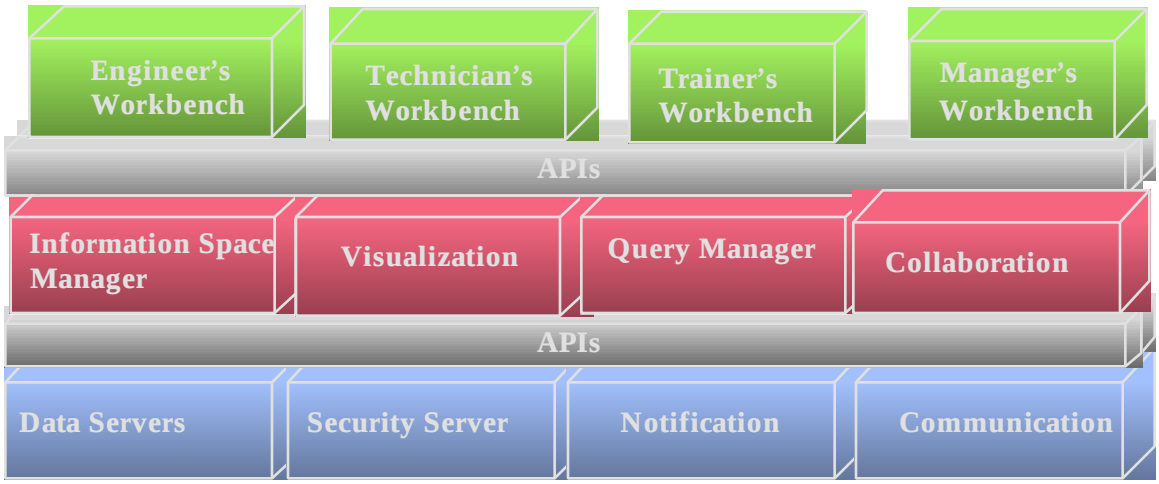


Figure 8 DIAMOND System Architecture

7. Hardware and Software Mapping

One of the main design goals of the DIAMOND project is to make use of commercial components. Development and deployment of the prototype will be based on cost effective and commercially available components.

7.1 Hardware Components

- Platforms:
  - Desktop PCs:
    - Windows NT
  - Mobile Computers:
    - Telxon's 1184 Portable (pen-based), Apple Newton 120
    - PC Laptop
- Wireless Connectivity:
- Communication within a Field Site: Spread spectrum: AT&T WaveLAN
- Communication between Sites: CDPD and T1.
- Wired communication: Ethernet
- Voice communication: Cellular Phone

7.2 Software Components

The software components were selected with the following requirements in mind:

- Reasonable purchase cost
- Low deployment cost
- Significant reduction in the development cost.
- Significant speed up in development time.
- Availability as a commercial off the shelf component
- Maintainability of the software
- Support

With these requirements in mind, we selected the following software components:

- User Interface: Netscape 2.0 and Java. Netscape and Java have not been around for a long time, but they look very promising and seem to exert already major influence on the way people are developing applications for the Internet. A very important feature of Netscape 2.0 is its capability for integrating applications with legacy user interfaces into a single coherent user interface design.
- Programming Languages: C++ and Java.

- Class Libraries: We selected Rogue Wave's set of class libraries. Rogue Wave provides three class libraries:
  - Tools.h++: A generic class library for C++ , offering datatypes such as containers, hash tables, linked lists and binary trees, reducing the development time for complex data types needed by DIAMOND.
  - Net.h++: a class library providing abstractions at the transport layer of ISO's OSI reference model. This allows the DIAMOND application programmer to concentrate on the higher levels of the protocol stack.
  - DBTools.h++: A class library that allows the creation of an interface that is independent of the actual database management system used. DBTools provides interfaces to several SQL compliant servers such as Oracle, Sybase and DB/2.
- Whiteboard system: Farsite (First candidate technology)
- Video Conferencing: TBD (confer with the KIWI project)

8. Global Resource Handling

The system provides user authentication, which determines whether a user is authorized to enter the information space. The authentication scheme is based on user name and password. A user is allowed to enter the information space if the correct name and password is given. The user interface for Login consists of a dialog box at start-up prompting the user to enter user name and password. The Logout dialog is entered when user clicks on 'quit' button in menu. The user has to click on the 'Yes' button to confirm he or she is logging out.

The system provides access control to documents. Depending on the security level of the document, it determines if a user is authorized to access the document, edit it or share it document with other users.

The system has a network-wide name server, which maintains a table of users currently logged on & their machine and location where they are logged on. The name server maintains a mapping between system servers and their respective locations in the system, allowing subsystem services to be changeable at runtime.

9. Data Management

The data management goals are twofold: Fast and consistent access to data in a mobile environment and uniform access to a set of heterogeneous data stored in different formats. Consistent access can be best provided by centralized data stores, because their is only once copy of any data item. In a mobile environment with limited bandwidth a centralized database management architecture does not always provide fast access. Fast access involves a

distributed database architecture with local copies of data for faster access. Local copies also provide the capability of accessing data during times of disconnected operations. However, many data items are updated and changed. For example, there are about 70 changes to EDMS drawings per month. Thus local caching requires a notification and invalidation mechanism. Uniform access means that we intend to provide access to the following databases:

- EDMS
- SIMS
- Vehicle Database: Sensor Data from the people movers
- Electronic Forms Database: Annotations, FSBs, Notifications, Work Orders

Figure 9 shows the overall picture of the database architecture.

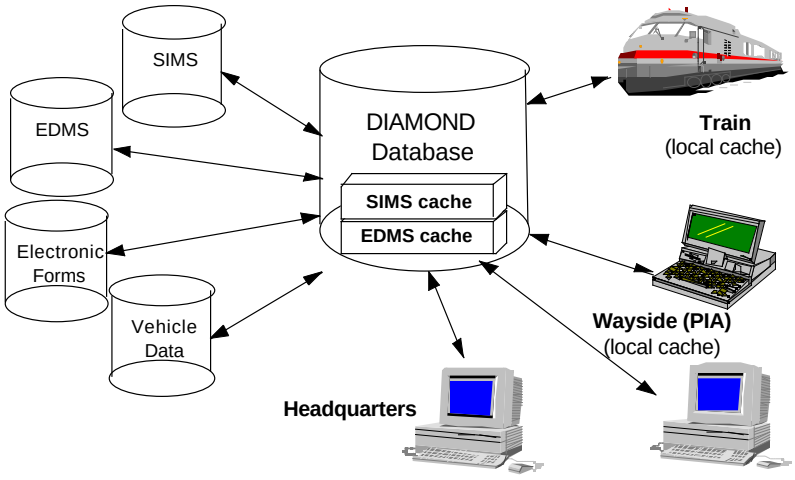


Figure 9 Database Architecture

The **DIAMOND Database** provides a consistent and uniform interface to each of these types of databases. consisting of four operations: Store(), Search(), Delete() and Retrieve(). Store() stores the (changed) document. Search() finds the data entries in the information space matching search criteria while Retrieve() gets the actual data selected from the searched document list. Delete() performs a Search() operation before actually deleting the data from the database.

The access to the individual databases (SIMS, EDMS, Electronic Forms and Vehicle Data) depends on whether the database is already part of the corporate information structure at Adtranz or a database that can designed from scratch as part of the DIAMOND project. Examples of legacy databases are EDMS and SIMS. Interfacing to these databases needs different solutions based on the accessibility or existing interface of the legacy system. For example, to interact with SIMS, the database interface must use an existing ASCII file format and

communicate via batch processing. To interact with EDMS involves interfacing to an existing software packages - WorkManager on top of ORACLE database. The system's database interface is shown in Figure 10.

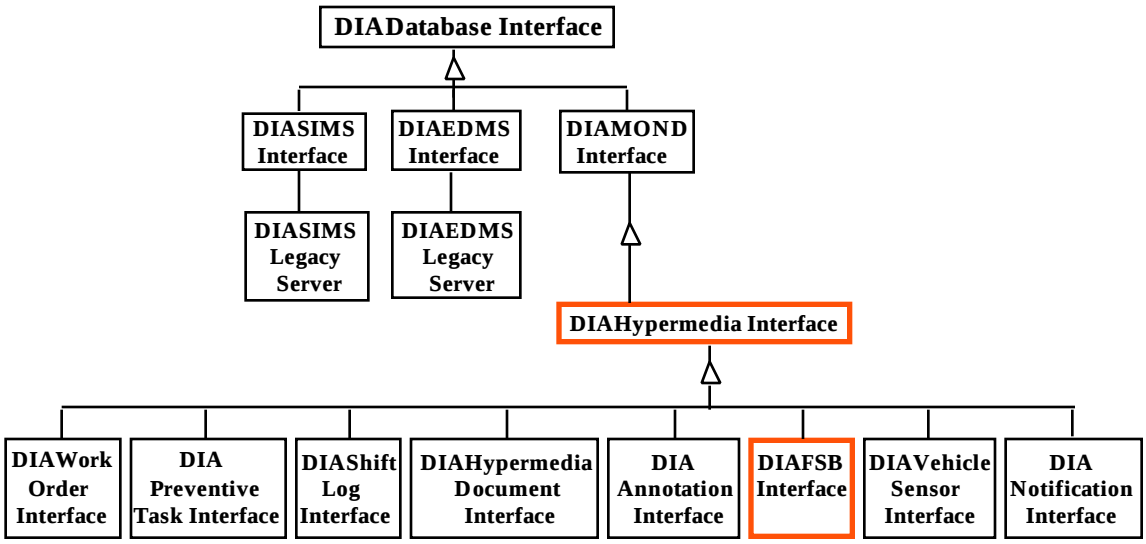


Figure 10 Database Interface to various DIAMOND databases

Another important aspect of the database architecture is its independence from the commercial database package used to provide the server. The Database Server shown in Figure 11 encapsulates the database interface from the server and provides plug and play capability with virtually any existing SQL compliant relational database management package.

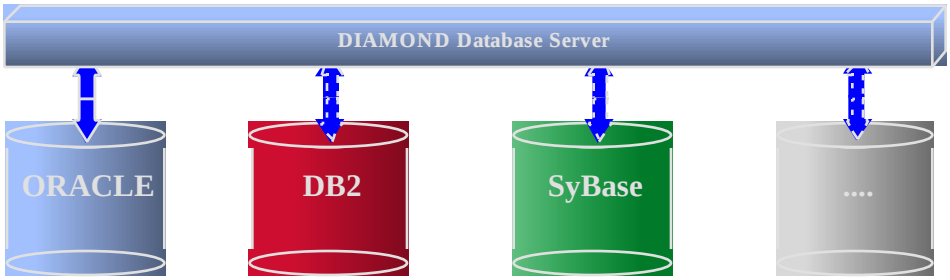


Figure 11 Database Server Interface

10. DIAMOND: Physical Connectivity

Diamond's physical connectivity is shown in Figure 12. The vehicle **ATO** (Automatic Train Operations) computer is a collection of onboard components hat collect the sensor data. The ATO is connected to the **Vehicle Data Server** via a RS-232 connection

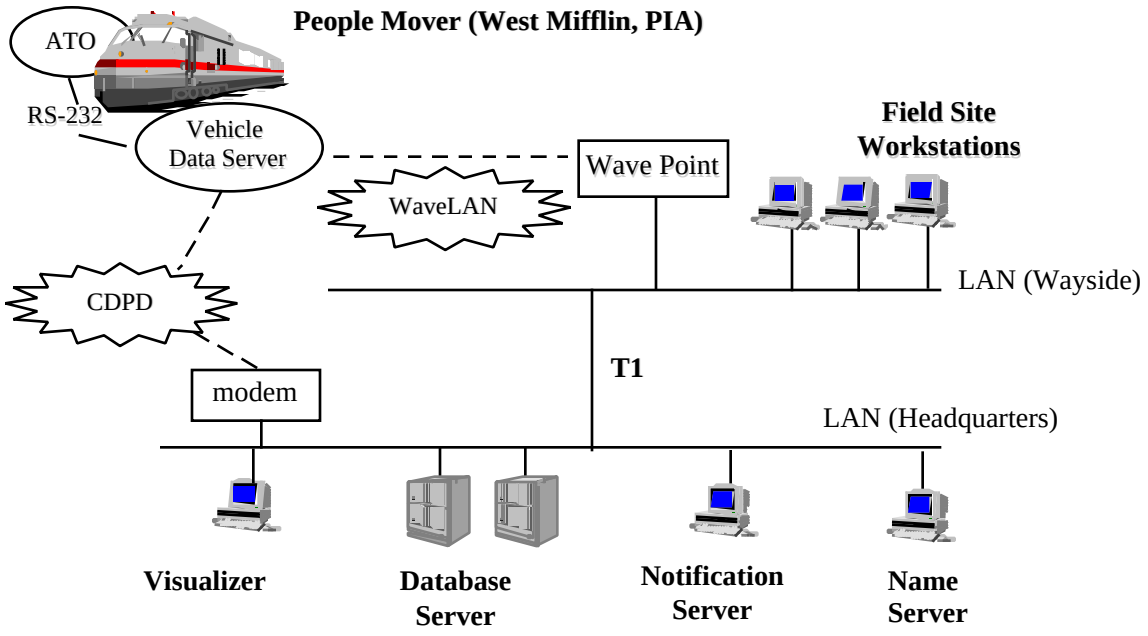


Figure 12 Physical Connectivity

The Vehicle Data Server has two types of wireless connections to the rest of the system. The connection to the Wayside is based on spread spectrum transmission (WaveLAN) to a base station attached to the local area network of the field site. Date is transferred from the base station to the field site workstation (e.g. DIAMONDTT at West Mifflin) where they can be visualized or stored locally. It can also be transferred via a T1 line to the local area network at **Headquarters**. At Headquarters the data can also be visualized or stored in the database.

The wireless connection directly to **Headquarters** is based on the digital cellular phone service CDPD (cellular digital packet data). This allow applications running on the People Mover to communicate directly with applications at Headquarters. While CDPD provides larger geographical coverage, its bandwidth of 19.2 bit/sec is significantly lower than the bandwidth offered by WaveLAN (1.6 MB/sec).

11. Boundary Conditions

The system deals with short term disconnections due to radio shadows without explicit interference by the end user or system administrator.

The system allows the users to continue to work in somehow limited form (no change notification, access only to cached local data) when they become disconnected from one or more database servers.

.