NASA/TM—2005-213415 AIAA–2004–6360

# Exploring Diverse Data Sets and Developing New Theories and Ideas With Project Integration Architecture

Theresa L. Benyo and William H. Jones
Glenn Research Center, Cleveland, Ohio

January 2005

The NASA STI Program Office . . . in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the Lead Center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA's counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.

- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or cosponsored by NASA.

- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results . . . even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA Access Help Desk at 301–621–0134

- Telephone the NASA Access Help Desk at 301–621–0390

- Write to:
  NASA Access Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076

NASA/TM—2005-213415 AIAA–2004–6360

# Exploring Diverse Data Sets and Developing New Theories and Ideas With Project Integration Architecture

Theresa L. Benyo and William H. Jones
Glenn Research Center, Cleveland, Ohio

Prepared for the
First Intelligent Systems Technical Conference
sponsored by the American Institute of Aeronautics and Astronautics
Chicago, Illinois, September 20–22, 2004

National Aeronautics and
Space Administration

Glenn Research Center

January 2005

## Acknowledgments

The authors wish to thank Kimberly Skinner, Entara Technologies Group, for figures 1 and 2 which helped illustrate traditional dataflow with the proposed data discovery framework.

Available from

NASA Center for Aerospace Information
7121 Standard Drive
Hanover, MD 21076

National Technical Information Service
5285 Port Royal Road
Springfield, VA 22100

Available electronically at http://gltrs.grc.nasa.gov

# Exploring Diverse Data Sets and Developing New Theories and Ideas with Project Integration Architecture

Theresa L. Benyo and William H. Jones
National Aeronautics and Space Administration
Glenn Research Center
Cleveland, Ohio 44135

The development of new ideas is the essence of scientific research. This is frequently done by developing models of physical processes and comparing model predictions with results from experiments. With models becoming ever more complex and data acquisition systems becoming more powerful, the researcher is burdened with 'wading through' data ranging in volume up to a level of many terabytes and beyond. These data often come from multiple, heterogeneous sources and usually the methods for searching through it are at or near the manual level. In addition, current documentation methods are generally limited to researchers' pen-and-paper style notebooks. Researchers may want to form constraint-based queries on a body of existing knowledge that is, itself, distributed over many different machines and environments and from the results of such queries then spawn additional queries, simulations, and data analyses in order to discover new insights into the problem being investigated. Currently, researchers are restricted to working within the boundaries of tools that are inefficient at probing current and legacy data to extend the knowledge of the problem at hand and reveal innovative and efficient solutions. A framework called the Project Integration Architecture is discussed that can address these desired functionalities.

## I. Introduction

A traditional workflow of data involved in a technological process is often slow and inaccurate since data needs to be constantly translated from one application to another (fig. 1). New architectures to enable rapid discovery of information patterns and correlations on an automatic basis (or, at least, an electronically-facilitated semi-automatic basis) are needed. The Project Integration Architecture (PIA) provides key technologies to facilitate the discovery process. PIA allows many researchers to quickly and usefully extend their knowledge base to discover complex relations, formulate new implications, and devise new methods of analysis within a vastly broader range of knowledge. Figure 2 illustrates how PIA can be used to facilitate rapid discovery.

PIA can satisfy and facilitate the following desires:

1) Provide a single environment in which all information from any source – static, dynamic, instrument, analysis, or archive – and of any level of complexity can be viewed, reviewed, manipulated, and shared.
2) Provide for the comprehensive, auditable, verifiable, and economical tracking of any given technological process. Such tracking can be used to constitute a legal record if so desired. PIA also provides the ability to record references to associated, but disjoint, elements of a process and can include not only the automatically generated information, but human generated notes, reports, sketches, and the like.
3) Provide a single environment in which information can be seamlessly transferred between experiments, analyses, and other elements participating in a technological process. Additionally, this environment provides a consistent information platform upon which agents (in the general sense of that word), constraint-based query engines, and the like may perform complex information gathering tasks.
4) Provide a framework in which domain-specific ontologies can be implemented. This ability provides a basis upon which high-level, natural-language, scenario-based queries can be formulated and carried out over a body of semantically-infused knowledge.

This paper will briefly describe the Project Integration Architecture (PIA) framework and its history. It will also describe in detail how PIA can address the 4 functionalities identified above to facilitate the exploration of diverse data sets and the discovery of new theories and ideas. The aggregation of the functionalities enables intelligent data management and the autonomous consumption of information.

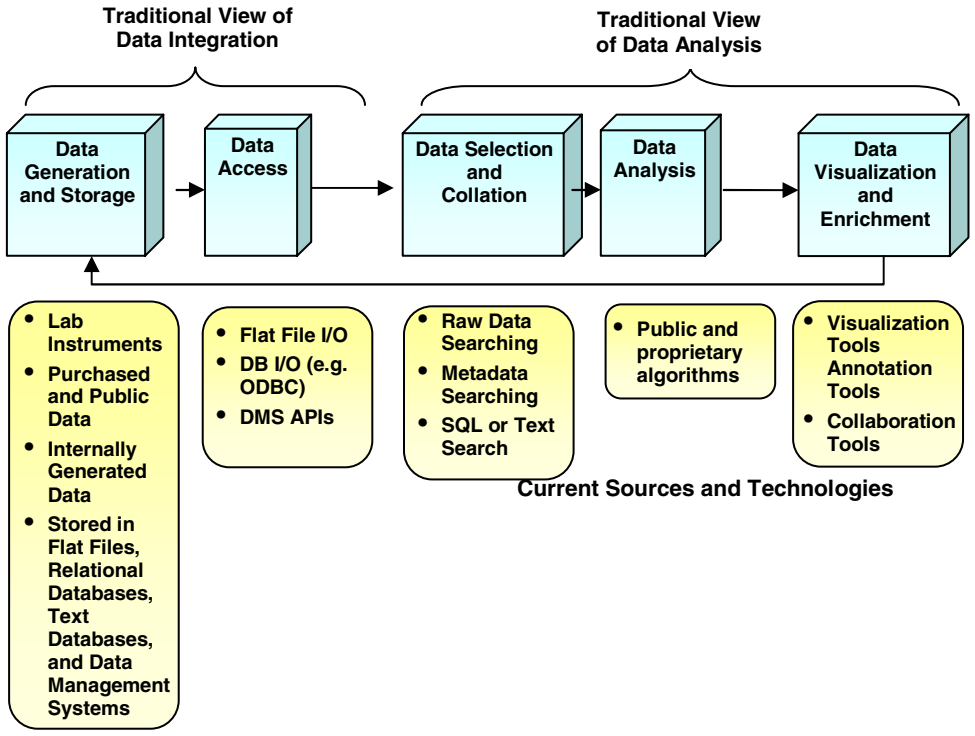


Figure 1.—Traditional workflow of data between applications in a technological process.

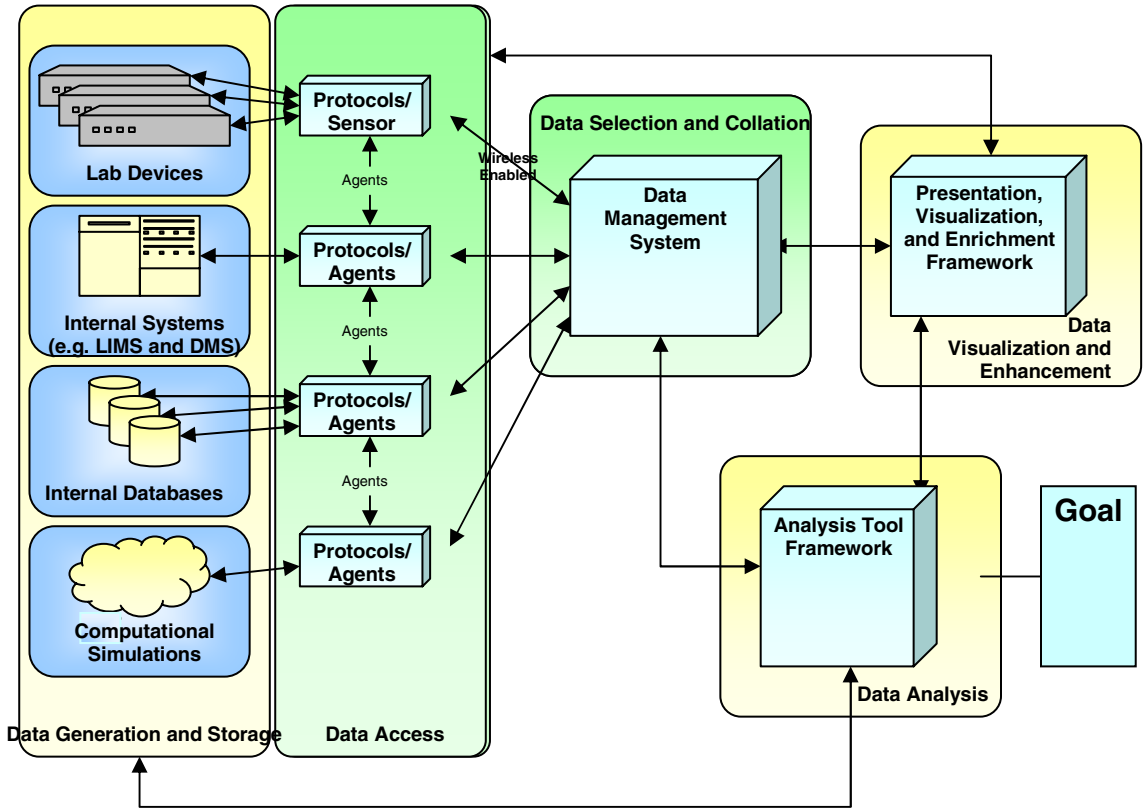


Figure 2.—The PIA framework facilitates data discovery in a technological process.

## II. Project Integration Architecture Background

### A. Brief History

In the late 1980s, the Integrated CFD and Experiments (ICE) project[1,2] was carried out with the goal of providing a single, graphical user interface (GUI) and data management environment for a variety of CFD codes and related experimental data. The intent of the ICE project was to ease the difficulties of interacting with these disparate information sources. The project was a success on a research basis; the basic concept of an integrated experimental and analysis environment was demonstrated. However, on review it was deemed inappropriate, due to various technical limitations, to advance the effort beyond the successes achieved.

A re-engineering of the project was initiated in 1996.[3] The effort was first renamed the Portable, Redesigned Integrated CFD and Experiments (PRICE) project and then, as the wide applicability of the concepts came to be appreciated, the Project Integration Architecture (PIA) project. The provision of a GUI as a project product was eliminated and attention was focused upon the application wrapping and integration architecture. One key re-engineering decision was made: the C language used by the ICE project was abandoned in favor of the C++ object-oriented extension to that language.

During the intervening years, work has proceeded and an operational demonstration of the PIA project has been achieved. The current effort has not achieved the complete range of functionality developed in the original ICE; however, the architectural dividing line has been more thoroughly defined and adherence to it has been more rigorous. Portability of the architecture, not only across platforms, but to distributed object architectures, has been demonstrated.

### B. Description

The Project Integration Architecture (PIA) is an internet-distributed, object-oriented, (relatively) platform-insensitive, software framework that provides in a manner intelligible to both man and machine for the seamless generation, organization, publication, integration, and consumption of all information involved in any process. While PIA principally contemplates technological processes, there is no element or restriction of PIA that would preclude the introduction of other processes, in particular business processes, to its environment. Thus, PIA provides for the encapsulation and capture of truly comprehensive processes.

Put in simple terms, the basic goal of the PIA effort is to capture in its entirety the usage of any technological process in a single, useful, well-defined form. This capturing is not limited to the simple output of the process itself, but further includes coordinating information and the technologist's own insights into the meaning of the information. The nature of a technological process is, by project design, nebulous: it is considered to be any computer-realized 'thing' which provides or generates useful information about a project. This may include geometry definitions extracted from Computer Aided Design (CAD) programs, geometry or other specifications derived from design code predictions, results of experimental investigations, analysis and simulation, and more. By bringing all of the information of the technological process into one architectural design, a number of advantages are expected to (and, it is believed, do) accrue.

1) The information about the information (referred to in some conceptions as the meta-data) is encapsulated with the information itself. Information about the conditions at which the information was generated, or the merit of the information, is no longer separately stored in a technologist's journal (which might eventually be lost).
2) A common tool set for the generic use of processes is possible. A single GUI with a single look-and-feel can be devised so as to reduce the technologist's learning curve for additional applications to that solely related to the application. The technologist's habits of exploring a problem can now be the same from one process to the next. If desired, other GUI's can be developed to handle special discipline-specific processes.
3) Common browsers and search engines may be implemented to peruse the supply of information in detail and convert it into information in general for end consumers of that knowledge.
4) By wrapping every application in a well-defined architecture, it is now possible to code into such applications the knowledge to acquire information automatically from other applications. Because of the architectural design, such coded knowledge is based upon the kind of information desired, rather than upon the application generating that kind of information.
5) Wrapped applications coded to obtain information based upon its kind may then be combined in a directed application graph to build, in effect, "super-applications". Applications of differing fidelities and disciplines may be mixed together as appropriate to the project. Figure 3 illustrates a "super-application". Each blue box represents an application wrapped within the PIA environment (as indicated by the application graph to

the left of the blue box). As a result, each application is linked together to form a "super-application" and can easily use needed information from each other.

6) The building of "super-applications" enables project-wide sensitivity analysis, genetic manipulation, optimization, and refinement to be conducted.

These advantages are further discussed in the following sections.

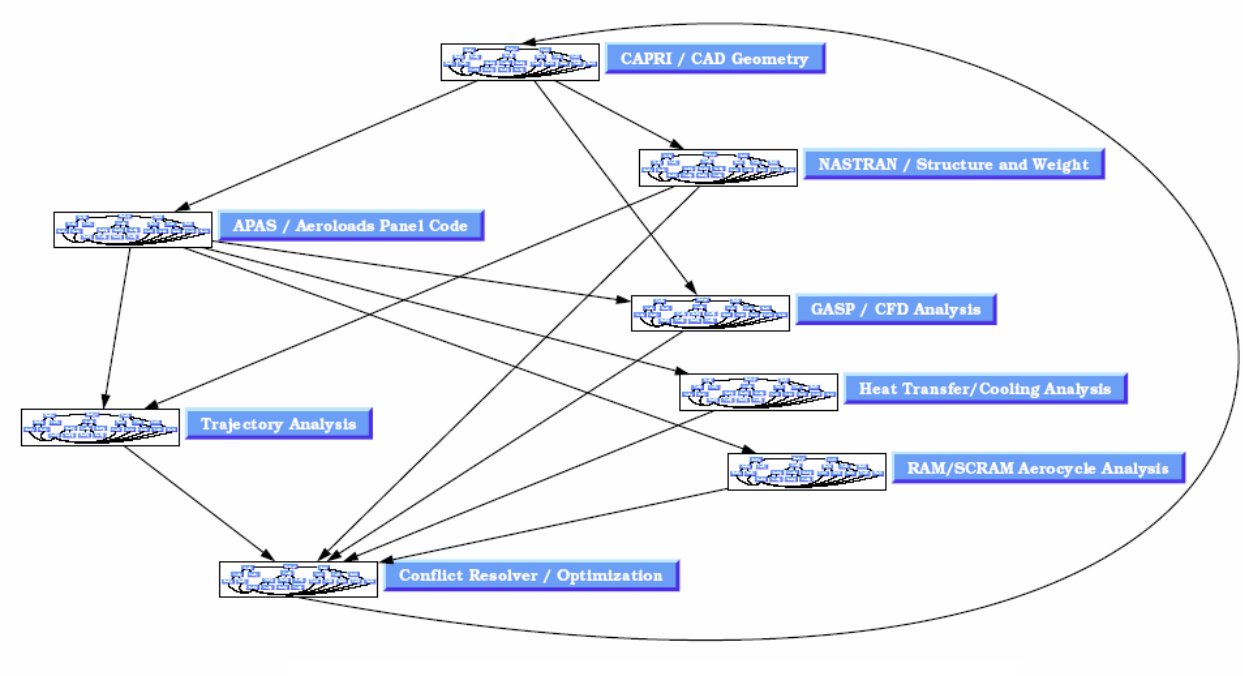


Figure 3.—An example of a "super-application" in PIA

## III. Exploring Data Sets

In order to explore large data sets, the elements containing the data must be infused with knowledge about that contained data. By achieving this self-awareness, all applications connected to the PIA framework can seek out its required data to successfully execute its task. For example, a CFD inlet simulation might need the geometry of the inlet to be modeled. If the researcher is interested in a certain type of inlet, the wrapper application can 'seek out' that type of inlet from the inlet geometries existing in the PIA framework.

### A. The Process Abstraction

PIA formulates the elements of a process – experimental results, design and analysis codes, simulations, rule sets, business models, and the like – on the most general and fundamental level. The reduction arrived at is that any such element is comprised of three parts: information input to the element, some transformation of that information by the element, and some useful output as a result of the efforts of that element. Further, the PIA abstraction states that any element of this reduction may be empty; that is, it is not required that there be any actual input, any actual transformation of information, nor any actual output (though it is hard to imagine the immediate utility of a resource with no output). This reduction allows virtually any resource – live instruments, data archives, running analyses; anything really – to be encapsulated with the facilities of PIA. Once so encapsulated, the information of each such resource is presented in a consistent manner which allows it to be easily identified, accessed, consumed, published, and integrated with other PIA-wrapped resources.

### B. Interacting with the Process Abstraction

Because of the great generality of PIA's process abstraction, no consumer of a PIA-presented resource is expected to approach a resource with any practical, specific foreknowledge of that resource. That is, it is not expected that a PIA compliant GUI would or should be devised for, say, the NASTRAN Finite Element Code with the appropriate expectations for what that code has and does. To accommodate this lack of practical, specific foreknowledge, PIA relies on the technology of self-revelation. Simply put, if one cannot know beforehand, one

must ask and respond appropriately. Thus, a GUI is built not with the knowledge that the application it is to access has structural finite element information, but rather is built to ask the application it encounters what information it, in fact, has and, upon discovering that it has structural finite element information, respond and interact with that kind of information appropriately. As experience with PIA development to date has demonstrated, this shift in paradigm does not incur a great implementation penalty, but only requires shift in implementation outlook and attitude.

It is expected that, to the extent that discipline-specific features are to be provided, many GUIs (as well as other entities) will be devised to work to the PIA standard as one size cannot be reasonably expected to fit all. Certainly generic entities may be devised since many discipline-specific forms of information are based on common representations of information; for example, the concept of a scalar or vector value mapped throughout a three-dimensional field is very common and one GUI can handle all such cases to the extent that only generic interactions with that information structuralization are needed.

## IV. Information Encapsulation, Documentation, and Retrieval

PIA provides a paradigm for information encapsulation, documentation, and retrieval which differs substantially from traditional methods such as relational data bases, structured files formats and access Application Program Interfaces (APIs), and the like. PIA begins this process by making a philosophical distinction: it encapsulates information rather than merely data. Physically, this distinction is meaningless since in any case a series of bit patterns is stored and an interpretation applied when that pattern is retrieved. The philosophical distinction is important, though, in that PIA seeks to store those bit patterns in a way that allows more than their simple interpretation as a real number, or an image, or as whatever. PIA's key intention is to allow man and machine to interpret a bit pattern in terms of its semantic use and placement within larger contexts. For example, a real number is not simply a real number, but is an upstream, far-field, initial-condition Mach number specifying a element of an associated flow-field solution, for an identifiable geometry, that is part of a larger air-vehicle system, and so on. By bringing this semantic richness to man and, more importantly, machine, data become more than merely numbers, it becomes useful information that can be interpreted and consumed in much more complicated ways.

### A. Information Encapsulation

PIA begins by defining an object-oriented ontology of all information. An ontology is a description (like a formal specification of a program) of the concepts and relationships (i.e., a vocabulary) that can exist for an agent (an object that can communicate and search for data and/or knowledge) or a community of agents. (Admittedly, the realization of this ontology is far from complete; but the pattern by which the ontology proceeds has been demonstrated.) By this ontological definition, a Mach number is encapsulated in not merely a real number interface (an interface being the equivalent of a C++ class), but in a Mach number interface. In so doing, a consumer of a Mach number, be it man or machine, can discover that the presented real number is, in fact, a Mach number. Furthermore, by dedicating an interface to each particular kind of information, self-knowledge of the encapsulated information can be infused into that interface. For example, the Mach number interface can "know" that its real number is dimensional in nature and that its particular dimensionality is non-dimensional. (Through the object-oriented technology of derivation and inheritance, dimensionality is, in fact, implemented at a derivational juncture common to all dimensional forms; thus, a pressure parameter uses the same dimensional implementation as a Mach number, but adds the distinction that its dimensionality is that of force per unit area.)

### B. The Descriptive System

Extending beyond this technology of semantic infusion through class derivation, all PIA application-level interfaces inherit a highly flexible, extensible, hierarchical system of self-description in which many other aspects of an encapsulated entity may be recorded as they occur. These descriptive elements include, but are not limited to, names, time stamps, sources of origination, change histories (including the identity of each entity, user or automaton, making each change), types, annotative texts, references to related documentation and the like, references to other particularly relevant parameters, descriptive journals, graphical sketches, digital signatures, measures of quality or reliability, access controls, and the like. The goal of the descriptive system in its design and implementation is to leave no piece of information lost in a file cabinet or archive box.

Perhaps the most important piece of the descriptive system is the introduction of a per-user, per-object access control system. The foundation layers of the PIA implementation introduced a distributed lock management system which provides traditional multiple-reader, single-writer style concurrent access conflict resolution. Through object-oriented slights of hand, the application level implementation of PIA extends this capability into an access control system, introducing the concept of a user rather than a mere holder of a lock and answering the question of not only

can this user read or write this object right now, but can he read or write it at all. This access control system was considered to be vital to any system containing information of actual value.

As mentioned above, the dimensionality of numbers is, at a particular derivational junction, infused into the ontological hierarchy. The information necessary to do this is, in fact, encapsulated in the descriptive system. While the implementation of this capability was nearly trivial, PIA-encapsulated numbers, when appropriate, know that they are dimensional in nature; they know the characteristics of their dimensionality (distance, velocity, acceleration, force, viscosity) and they know the units in which they are measured (furlongs per fortnight squared, for example). Dimensionally-aware parameters refuse to be accessed in a non-dimensionally sensitive manner: the number may be 12.0, but it is impossible to simply get that value; one must ask for the value in some dimensionally-aware manner, in which case the parameter will provide the value 12.0 inches, 1.0 feet, or 30.48 centimeters. Additionally, dimensionally-aware PIA parameters protect their dimensional characteristics: as it is impossible to divide a length by a time and assign it to a force; such a result may only be assigned to a velocity.

### C. Ecdysiastical Sorting

Another form of information documentation provided by PIA is an ecdysiastical sorting of the information bearing objects within each instance of an application. This sorting sorts information-bearing objects not simply by their apparent or surface kind, but by all their inherited kinds. For example, an upstream, far-field Mach number is sorted not just as an upstream, far-field Mach number, but as a far-field Mach number, a Mach number, a dimensional number that is, in fact, non-dimensional, a dimensional number, a number, a parameter of an application, and so on. This sorting allows information to be located at the semantic level of the searching entity. This is necessary, ultimately, because most applications are expected to derive their parameters (and other objects) beyond the defined, well-known levels of the domain-specific, ontological libraries. Though necessary for that specific reason, the sorting is useful for those situations in which ultimate semantic specificity is not necessary. For example, a search for all occurrences of hypersonic information would not necessarily care whether the Mach numbers were upstream, downstream, far-field, or near-field: it would simply want Mach number information of any kind, which this sorting would take it directly to. Figure 4 illustrates ecdysiastical sorting with a upstream far-field gas total pressure parameter.

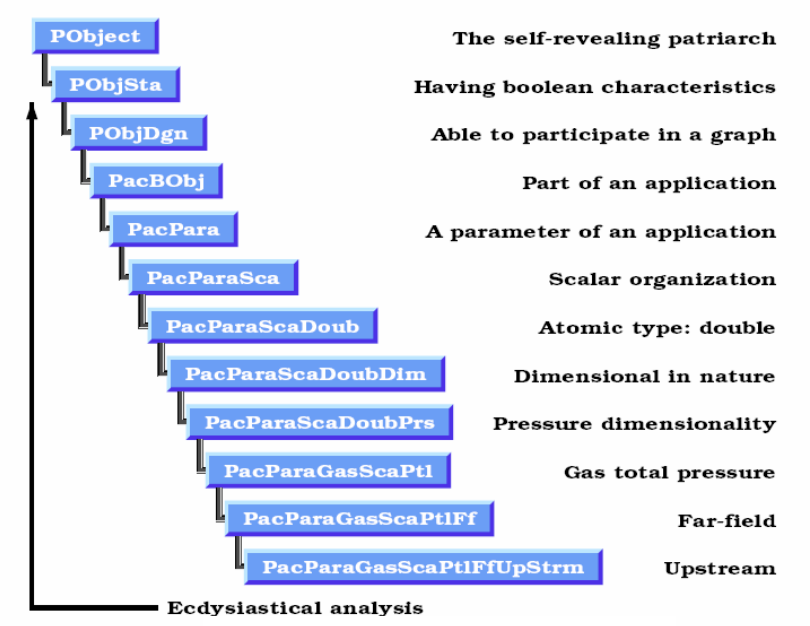


Figure 4.—Ecdysiastical Sorting in PIA

### D. Tracking Technological Processes

As part of its path-of-investigation-tracking goal, PIA records each distinct state of an overall process in a separate “configuration” object. All parametric information encapsulated within PIA, at present, exists within such a configuration. Another level of information documentation is introduced by this design since all information within a given configuration is considered to be related in that it specifies a part of that process configuration. This discards

the old data/meta-data paradigm in that all information within a configuration exists on an equal footing: whatever information item is considered to be the “data” is surrounded by its “meta-data”.

Furthering the configuration level of information documentation, PIA allows wrapped applications to be assembled into directed graphs of many applications cooperating toward a single goal. Configurations are kept in rigid synchronization within such application graphs and, thus, it can be inferred that information within corresponding configurations constitutes a yet larger whole. Each configuration operates within the confines of the defined operations for that application. Figure 5 illustrates how a PIA application consists of three main tree structures; an operation tree, a configuration tree, and an identification tree. Extending along another branch of such reasoning, applications, whether in an application graph or not, have within PIA the facilities to record the locations from which information for the operation was obtained. For example, a flow code that searches a PIA-presented wind tunnel archive for a flow field similar to the one posed to the code for use as a starting point can record a reference to that archived resource; it can then be inferred that the information of that wind tunnel archive element relates to the information of that flow code configuration, though possibly somewhat less so.

PIA process configurations are kept in an *n-ary* tree structure designed to further record the path of investigation. At any given point in an investigation, all the possibilities examined can be represented as siblings within that tree. When some few configurations are selected as likely possibilities, further investigations are represented as descendents of those tree elements. (As a mere economy, descendent configurations need only contain those information elements that distinguish them from their heritage; missing elements are inherited from the nearest ancestor in the direct lineage containing the needed element.) In saying that a particular information object is related to all the information objects of its configuration, one is actually saying that it is related to all the information objects actually inherited by that configuration. Furthering this, though, it may also be said that an information object is also somewhat related to all those information objects in its lineage beyond that which it inherits.

As a consequence of the inheritance of information within the configuration tree, PIA has moved the job of identification to another structure, the identification tree. While the configuration tree may expand without limit as the path of investigation leads, the identification tree exists only once for each instance of an application and is specific to the particulars of the application. The identification tree serves to represent the structuralization of information within the application and, thus, allows information within the configurations to be flattened. It is within the identification tree that yet another form of information documentation may be found since it may be inferred that smaller spans within the identification tree correspond to closer relationships between the identified elements of information. This is particularly true in that, in its Internet-distributed form, PIA has introduced the concept of indefinition within the identification tree; that is, a single element of the identification tree (possibly heading of sub-graph of that tree) may consider itself to be repeated an indefinite number of times, that number probably being specified within the configurations of the application. This feature was devised to allow for applications, such a multi-block flow codes, which allow the indefinite repetition of data structures within the application. Within such an indefinite identification, the implicit span between identifications becomes highly relevant to the inferred relationships between informational elements. For example, identifications from different repetitions relate less to one another than if the identifications were of the same repetition.

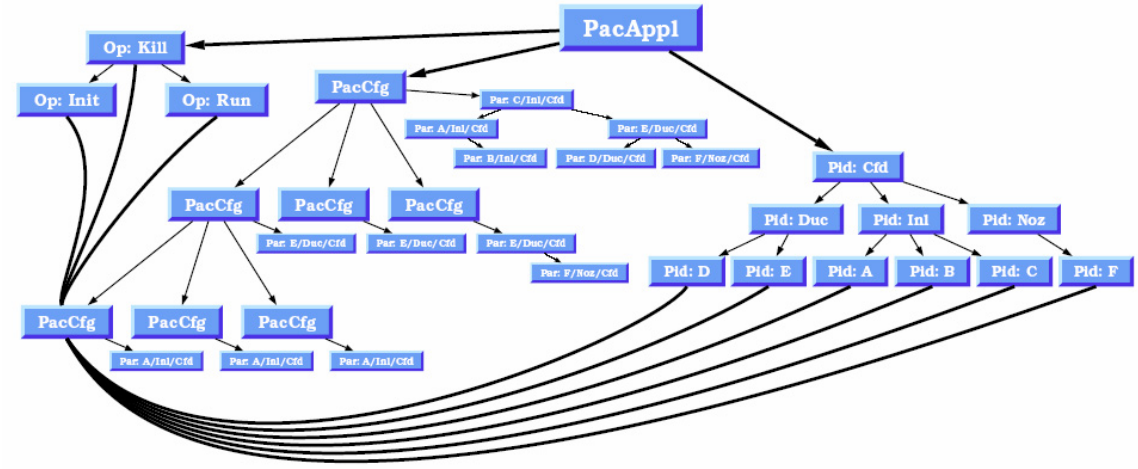


Figure 5.—The PIA Self-Revealing Application Architecture. *The figure shows each of the three tree structures from left to right; operations tree, configuration tree, and identification tree.*

## V. Information Transfer

PIA uses the technologies of self-revelation and semantic infusion through class derivation described in previous sections to enable the seamless flow of information throughout a technological process.

### A. Transfer of Information through an Application Graph

The primary method of arranging information flow throughout a technological process within the PIA framework is to arrange cooperating processes into a graph. (As used by PIA, a graph is an information structure having a single initial node, edges connecting that initial node to successor nodes, edges connecting those successor nodes to more successors, and so on. Except for the initial node, the general interconnection of a graph is on a many-to-many basis.) The expectation of the process graph is that information flows from predecessors to successors. Thus, as each node of a process graph comes to a state of completion, it informs each of its successors in the process graph of that fact and allows them to inspect it for information that might be of value to them in their expected operation.

### B. Acquisition of Information from Non-Participatory Repositories

It is not certain that every process participating in such a process will necessarily find within the confines of the process graph in which it is participating all of the information it needs to operate. For example, some analysis codes may need an initial starting point from which to begin their computations which is not explicitly contained by the process graph. In such cases, the process is free to explore the wider range of resources provided by the PIA environment in order to locate such information. PIA provides descriptive elements appropriate for noting the source of such information in the event that it is located.

### C. Acquisition of Information by External Consumers through Constraint-Based Queries

Using PIA as an information foundation, it is then possible to build query engines that can search the PIA-published knowledge base to locate specific kinds of information. For example, a researcher might wish to explore the known unstart conditions for a particular kind of inlet in order to identify appropriate startup conditions for an experiment using that kind of inlet. A query engine could then be told to look for instances of the specified inlet geometry that associates a supersonic flow solution with that inlet. The query engine would be devised with (or accept as a search specification) knowledge about the characteristics of an unstarted flow and would respond with those identified flows having those characteristics.

As with processes in an application graph, such query engines would be programmed to seek the kind of information needed (using the technologies of self-revelation and semantic infusion) rather than to seek a specific, pre-programmed source for that information. As with the example above, the query information would search for inlet geometries and associated flow-field solutions, not for the output files of known Computational Fluid Dynamics (CFD) programs.

## VI. Reliability and Scalability of the Implementation

### A. The Object Life Cycle

All PIA objects persist from their creation to their explicit destruction, without regard to whether or not their server or servers are continuously in operation. (Uncontrolled server crashes may still have unfortunate results, but possible workarounds to even this eventuality have already been proposed.) The PIA implementation has code hooks (which will be exploited in the near term) to provide for redundant, automatic-failover, multi-server object service, enabling highly-reliable information resources. Furthering that capability, PIA also provides for redundant archival storage of object states and, anticipating commercial concerns, the remote storage of information that might be considered proprietary (or otherwise sensitive) in nature.

The PIA object-life-cycle paradigm was explicitly designed in the expectation that the vast majority of objects would be inactive at any time. Such inactive objects are flushed from their present server, their internal state being recorded on one or more archival stores (as suggested in the preceding paragraph). Should a method be delivered for an inactive object, the object is recreated in its server (or, in the case of a redundant server cluster, one of its possible servers), its internal state is restored, and the method finally delivered to the object for execution and response. The object life-cycle paradigm and its implementation allow PIA to seriously contemplate information volumes extending beyond terabytes, petabytes, and exabytes to the practical infinity.

**B. Distribution of the Implementation**

PIA is built to run on a distributed network of servers using the Common Object Request Broker Architecture (CORBA) standard defined by the Object Management Group (OMG). By building upon the CORBA foundation, objects actually implemented by one physical machine on a network may be accessed by any other machine able to reach that implementing machine over that network.

The CORBA standard defines only basic mechanisms for distributed object interactions. Many particulars are left as selections or, in some cases, inventions of the particular technological process built upon the standard. For example, the particulars of object creation and destruction are not specified by the standard, but are left to the details of the technological process involved; some technological processes may serve a fixed set of objects for the entire lifetime of the process while others may create and destroy objects moment by moment.

Using the flexibility of the CORBA standard, PIA defines processes for the creation and destruction of served objects. For purposes yet to be discussed, PIA also defines two levels of association between compliant servers of objects: a greater level in which servers join to form a collective providing many diverse technological processes, services, information resources, and the like and a lesser level in which two or more physical machines join to form a highly-reliable cluster of servers serving one or more processes or other resources.

Because PIA builds upon the CORBA standard and because of the association of servers into clusters and of clusters and servers into a collective, PIA provides for the geographical dispersion of technological processes. Further, excepting the limitations of network firewalls and the like, anyone or anything with a PIA-compliant access portal on a network able to reach the network of a PIA collective is able to access and utilize all the resources of that collective (subject to imposed access controls, of course). Figure 3 can also be used to illustrate the distribution of servers in a PIA environment. Each application shown in the figure can exist on a separate server residing on a network of servers controlled by CORBA.

The formation of PIA-compliant application level servers into a cluster brings the resources of (potentially) many machines to bear upon a single PIA-served application or other information resource. By using various distribution methods, the objects implementing an instance of such a resource may be spread over the physical machines of such a cluster, thus bringing greater computational power and storage resources to a situation that might be more than any single machine could reasonably handle.

**C. The Cluster as a Reliable Server**

The PIA cluster offers a further benefit: any given object may be served by more than one member of a cluster. Should one or more machines of a cluster be halted (preferably in an orderly manner), the PIA implementation allows a consumer of a given object to obtain access to that object through some other member of the cluster. Typically, an object is set up at the time of its creation to identify particular machines of the cluster able to serve that object; if any one of those machines is available, then the object should be available. This does leave open the possibility that all the machines identified by a given object might be down at the same time; however, the probability of such an occurrence can be calculated given the particulars of a cluster configuration and the configuration adjusted so as to give an acceptably low result.

**D. Formation of Servers and Clusters into a Collective**

After the benefits of clustering are realized, servers and clusters may be joined to form a collective. The implementation of PIA is such that the entire resources of a given collective are available to any accessor throughout the collective. PIA imposes no practical limits as to the range or extent of a collective – in theory a collective could extend to include every computer connected to the Internet. The PIA implementation is built upon a CORBA implementation that supports at least one encrypted communications protocol so that even such extra-PIA issues as network communications security might be successfully addressed.

Bringing all of the computers on earth into the PIA world would be of little benefit if there were some real limit on the number of objects PIA could identify. Because of this, PIA utilizes the flexibility of the CORBA standard to make for itself a virtually limitless object “address space”. In the CORBA world, every object is uniquely identified. At the CORBA application’s selection (where PIA, itself, is the CORBA application in this case) this identification may either be generated automatically by internal CORBA services, or it may be generated by the application. The PIA implementation has chosen to generate those object identifications for itself and includes in those identifications the kind of object created, the time of object creation, the Internet Protocol (IP) address (including the new Version 6 address form) of the machine creating the object, the process identifier of the program creating the object (which is unique within a given machine), and a unique number within the program creating the object. As implemented by PIA, these object identification factors combine so as to allow, currently, every process of every machine on the Internet to create 16 billion, billion objects of every kind served by PIA every second for the next 500 billion years

before a redundant identification would be generated. Furthermore, should that identification space prove insufficient, it may be extended by simply shutting down the appropriate servers, providing amended code, and resuming operations.

Having an object identification space that realizes a practical infinity does not entirely specify the size of the information space; it is necessary to know how big each object is before that value is available. In the case of PIA, objects may vary in size from very small to very large. As with many other things, moderation in object size is probably best. Objects that are too small (in terms of actual information content) will certainly incur significant overhead penalties while objects that are too big will probably be inconvenient for the serving program to handle.

## VII. Developing New Theories and Ideas

The advancement of any technology relies on the development of new theories and ideas. One way to do this is to explore an existing base of knowledge to discover what is NOT there. Another way is to query a database of knowledge with a "what-if" question. On the flip side, one wants to avoid re-inventing the wheel if such technology already exists. PIA can address data discovery by taking care of the mundane bookkeeping required with massive amounts of data and work with intelligent agents and ontologies designed to help answer "what-if" predictive questions.

### A. Leave the Bookkeeping to PIA

The traditional role of the computer is as a tool under the more or less direct command of a person. Typically, a person prepares some sort of input, directs a particular program in a computer to process that input, and then examines the results produced. In this paradigm, the person is very close to a direct causal element in the production of information while the computer is very little more than a highly sophisticated tool. The relationship between man and machine is very close to that of a laborer feeding sheet metal blanks into a stamping press and plucking the formed parts out of that press.

Perhaps the most far-reaching goal of PIA was to elevate the roles of man and machine, raising the machine to the level of the laboring man while the man is raised to the level of reflective supervisor and evaluator. Much of this goal has been realized through the technologies elucidated in the previous sections which allow the machine to take over in a practical sense the more mundane activities of complex technological processes; the location, recognition, and error-free transport of information, the meticulous bookkeeping of process configurations, the generation of auditable trails, and the like. The last phase of the elevation of the machine will now be discussed: the phase in which the machine takes over not just the operation of the technological process, but the formulation of that process as well.

PIA is certainly not the only effort that integrates technological processes; indeed, there are already a number of commercial products available in the marketplace that provides some measure of solution in this area. The distinguishing feature of all of these products, though, is that they all still rely upon a person to find and connect the dots of the technological process to be executed. Unfortunately, experience with these products has shown that the practical ability of a person to connect the dots, even with a reasonably sophisticated computer-implemented tool, is limited and reached with relative speed and ease. About 15 to 20 applications represent the practical limit.

By once again re-applying the technologies of self-revelation and semantic infusion through class derivation, PIA is able to turn the task of technological process formulation over to the machine. With the mindless inerrancy of the machine, integrations of technological processes involving hundreds and thousands of applications ought to be within easy reach.

One basic formulation of this feat is to pose the technological problem in terms of an optimal result of some identified parameter or combination of parameters. This is the classic optimization statement: get the best value of this parameter (that is, objective function) that you can. In optimization, though, the process of obtaining the objective function from an independent design vector must be handed to the process as a *fait accompli*; but with PIA that process can be left as an open issue to be resolved.

PIA enables the resolution of the objective function issued through the technology of self-revelation: it allows the generator of the objective function process to search the range of all applications, inquiring of each whether or not it produces the desired parametric result. Assuming that some application does, in fact, produce that result, the generator inquires of that application as to the set of inputs it needs to produce the result. Figure 6 illustrates this condition with a small example of how 'Application A' depends on 'Application B' and 'Application C' which provide the parameters 'Application A' needs to produce its output of cost/pound. These cited inputs then cause another round of searching to identify the applications that might produce those needed inputs. This generator

process of finding applications that produce the currently needed information elements proceeds on iteratively until at last the set of still-needed inputs reaches a state at which they may be regarded as an independent design vector.

A fundamental presumption of the currently foreseen algorithm is that the application graph being developed by the solution generator process will be acyclic; that is, no application will require as its input an output of that same application, or of a successor to that application. Under this stipulation, information flow will be strictly top-down in its nature. It should be noted, though, that this presumption is not considered to be a permanent limitation, but only a current limitation for foreseeable research. Ultimately, it should be possible to handle cyclic information flows, possibly by wrapping the cyclic portion within a "solver" application. This would isolate the iterative portion of the solution process and maintain the overall appearance of strictly top-down information flow.

It is anticipated in future work that PIA will inform the user about the infinite loop and cease further operations. (PIA presently implements an interrogative for parameter objects identifying them as candidates for independent design vectors. Whether an affirmative response to that query should be taken at face value, or only accepted after a consistent failure to find an appropriate product from some other application, is a matter yet for research.)

At this point, the formulation of the technological process to produce the objective function value is complete. The applications identified are arranged into an application graph reflecting the flow of information established. The parameters of the identified independent design vector may be fed into the initial node of the application graph and, after the act of information propagation, the value of the objective function flows out of the terminal application node of that graph. This technological process may now be turned over to an optimization entity for exercise and production of the best obtainable result.

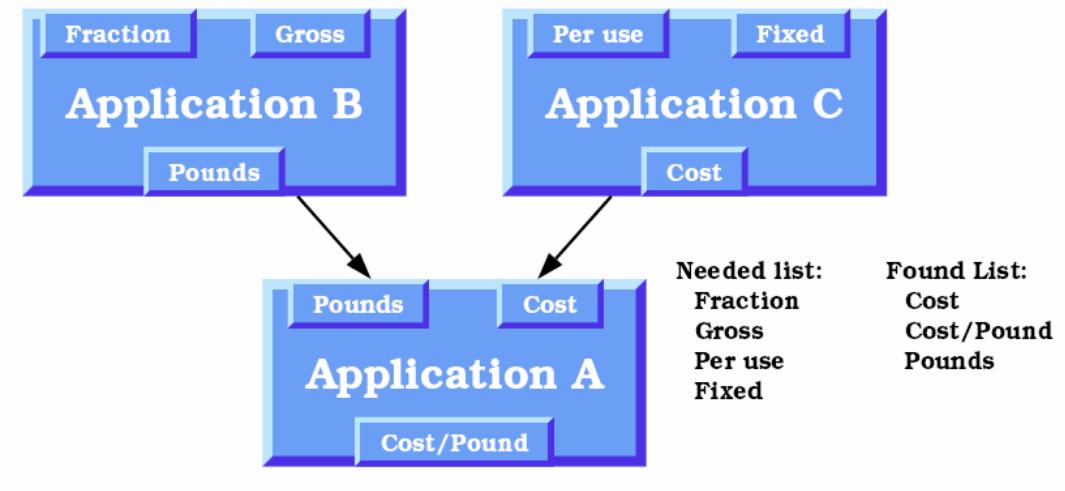


Figure 6—Propagation of information in an application graph.

The optimization entity, as with the generator of the technological process, is not expected to be an explicit part of PIA; rather, they are both expected to be consumers of PIA capabilities and services. The optimization entity, having been handed the defined technological process, begins to operate, probably by inquiring of the applications and parameters as to their nature, the existence of constraint boundaries, and the like. Figure 7 illustrates an example of this optimization functionality. 'Application X' only needs a random design vector to start the optimization process. As the results propagate through the PIA wrapped applications, a result is obtained. This result then goes through the optimization entity (created with some or all of the phases of operation as listed below) and determines whether another run through the applications is needed. The overall methodology of producing an optimal result for the defined technological process may involve some or all of the following phases of operation.

1) **Statistical Characterization**: The various parametric elements of the independent design vector may be systematically varied in order to establish which elements significantly affect the objective function value obtained and which are relatively ineffectual. Since the size of the independent design vector has a great bearing on the amount of work to be done in establishing the optimal result, reducing the size of that vector by eliminating (at least for a time) those ineffectual parameters is of considerable interest.
2) **Problem Partitioning**: Considerable research has been and is being done on partitioning large optimization problems into appropriately interrelated groupings which may be optimized, at least for a time, as individual entities. Such partitioning greatly reduces the level of effort for large problems. The concern,

though, is to assure that the optimization of the individual groupings does indeed tend toward an optimal whole. It is hoped that PIA can provide, through self-revelation, the answers necessary to identify such partitioning on an autonomous basis.

3) **Genetic Manipulation**: Large, complex systems often have many local optima which tend to defeat the usefulness of classic optimization techniques. Genetic manipulation technologies in which large, systematic variations of the independent design vector are made have often demonstrated the ability to sift through the set of local optima and correctly identify the global optimum. Interestingly, genetic manipulation technologies have been observed to identify global optima whose design points are wildly at odds with conventional expectations and experience.
4) **Classic Optimization**: Once the approximate location of the global optimum is established, the various classic optimization techniques such as conjugate gradient, method of steepest descents, and the like can be applied.
5) **Six-Sigma Technologies**: Often truly optimal results closely approach one or more design constraints. Variations of real manufacturing processes may cause the design point achieved to slip over one or another such constraint. Six-sigma technologies exist that allow such proximities to be identified and the design point backed slightly away from the ideal optimum to provide some real margin for error.
6) **Uncertainty Analysis**: In a manner somewhat akin to six-sigma analysis, uncertainty analysis seeks to determine the likely range of output values given some measure of uncertainty in the inputs to the technological process. These results may be used to establish some level of confidence in achieving those output results.
7) **Automated Risk Assessment**: There is yet another facility that PIA has implemented: the ability to associate a *beauty* with a parameter, a parameter configuration, or other object. Simply put, beauty measures the merit of the described object on a rather arbitrary scale: a value less than zero indicates that the object is wrong in some way, a value of zero indicates that the object is without merit but is not wrong, a value of unity indicates that the object is just a random thing out of the ether, and, finally, bigger values of beauty are better than smaller values. It is believed that PIA applications may be certified for their ability to improve beauty as they transform inputs into outputs. Further, it is expected that this improvement can be dynamic in nature. For example, a computational fluid mechanics application may conclude that its delivered improvement in beauty is inversely proportional to its residual values; residuals approaching zero will cause the application to produce a greater improvement in beauty while residuals that do not approach zero will cause the application to produce a lesser improvement or may even reduce beauty. Using this concept of beauty throughout the exploration of the technological process, it would seem possible to identify areas of the problem space in which beauty does not greatly improve and, consequently, judge those areas as at more risk than areas of the space where beauty does reliably and significantly improve.

When all of this comes to pass, the machine will certainly have been elevated to the level of laboring technologist: having had a problem posed to it, it will have devised a method of solving that problem and exercised that method in a search for the best value that can be obtained. Once this has been achieved, it will be appropriate to examine the role of the man. What does the technologist displaced by the laboring machine do?

The answer to that question is quite simple: the technologist displaced by the laboring machine now critically reviews the results of those labors. Based upon the learning and experience of the human, based upon the factors still outside the encapsulated knowledge of the machine, are the results produced by the machine believable? Do the results make sense or has the machine revealed a flaw in the knowledge it was taught? If no flaw is apparent, but the result seems radical in nature, what is the path by which the machine came to this conclusion?

Remember that the technologists, in reviewing the labors of the machine will have at their disposal not simply a final answer, but the entire trail of exploration that led the machine to its conclusion. The technologists will have both the process by which the technological process was arrived at and the path by which that technological process was exercised. All of the mechanisms of tracking, configuration control, description, rationale capture, and the like provided by PIA will be at the disposal of the solution and optimization entities utilizing PIA's capabilities and, since those entities are themselves simply more mindless, machine-implemented automatons, they should have no hesitancy in fully exercising those mechanisms to document their efforts.

### B. "What-if" and Predictive Questions

PIA intends to serve only as a common foundation for the representation of processes and the information contained within, generated by, and consumed by those processes. As illustrated by the preceding sections, PIA offers many forms of information description, relation, inference, and implication that may be exploited in an enormous number of ways, giving rise to many different approaches to information query, mining, and inference

automatons. Because of this great breadth of query systems, forms, algorithms, and the like, PIA does not propose to implement such capabilities itself, or at most PIA will provide only skeletal implementations of such services. Intelligent agents can be built on top of PIA's information base to enable "what-if" studies on information, topics, research areas, etc. that are already a part of the PIA information management infrastructure.

Several groups have explored the topic of intelligent agents and ontologies within a computer science framework. Gorodetski, et al.[5] starts with a JDL data fusion model and evolves it into a more detailed architecture. They propose that meta-models of distributed data sources and classification schemes can only be solved successfully through the thorough development of a distributed application ontology. In Reinoso Castillo, et al.,[6] an Intelligent Data Understanding System (INDUS) is described that is based on a federated architecture of distributed data sources. The INDUS project does not specifically call out autonomous agent technology as a key component, although their instantiator concept is similar in function to an agent. A query-centric approach, defined in terms of the individual data sources' local ontologies is used. This approach is based upon a global ontology and offers great flexibility when developing user interfaces to the data sources.

There are various definitions for agent. A commonly used one is the following,[7] which states: "… an agent is a proactive software component that interfaces with its environment and other agents as a surrogate for its user, and reacts to changes in the environment."

Several characteristics of an agent are that they are autonomous, adaptable, knowledgeable, mobile, collaborative and persistent. Through the technology of self-revelation and semantic infusion through class derivation, PIA provides a base of information upon which such agents may act. An agent architecture can be devised and implemented that would allow a particular instance of an agent to cruise from PIA server to server, browsing over the offered information and capabilities. Such an agent can gather needed bits and pieces or that agent might simply note where a particular piece or pattern was found. Further, an agent needing information of a particular kind or pattern might identify PIA-served applications that generate such information and either exercise those applications at the time they are found, or note their location so that the wrapped application might be exercised at a later time. This approach would be useful in mapping global ontologies provided by users into local ontologies generated by PIA agents.

In addition to providing the base upon which an agent architecture might be built, PIA exhibits some of these agent characteristics in its application wrappers. Application wrappers incorporate specific knowledge about the needs of the applications they wrap. A wrapper adapts to the cooperating-application environment in which it finds itself as determined through the technology of self-revelation. The wrapper must be coded, through one means or another, to draw upon the information it finds to be at hand in order to meet the needs of the application it wraps.

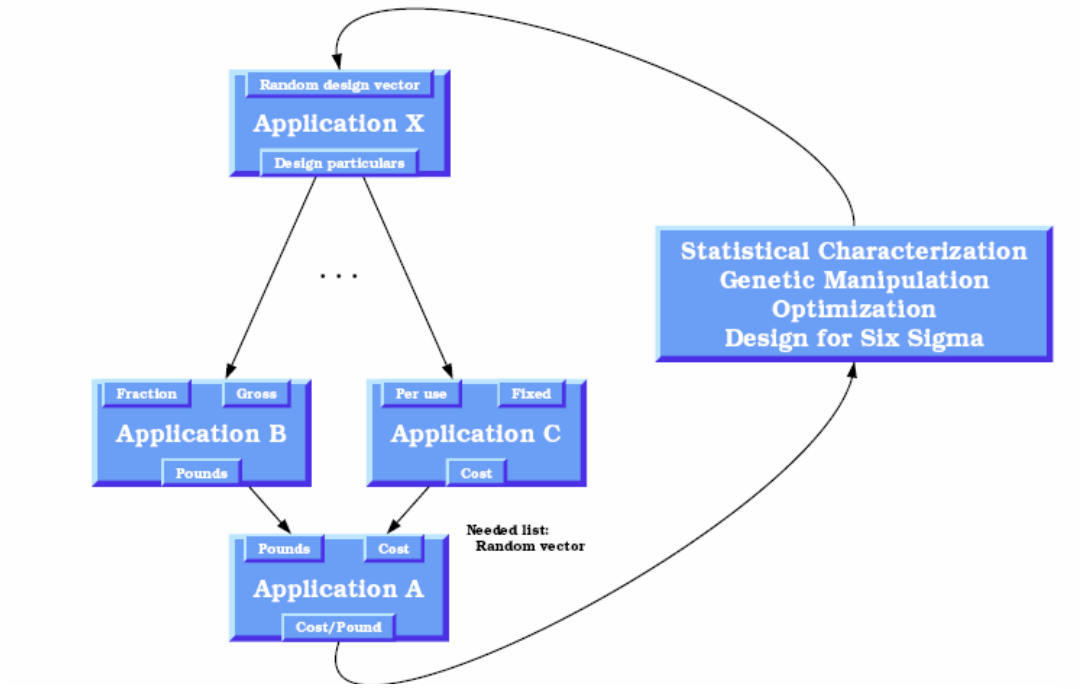


Figure 7.—Optimization Functionality in PIA

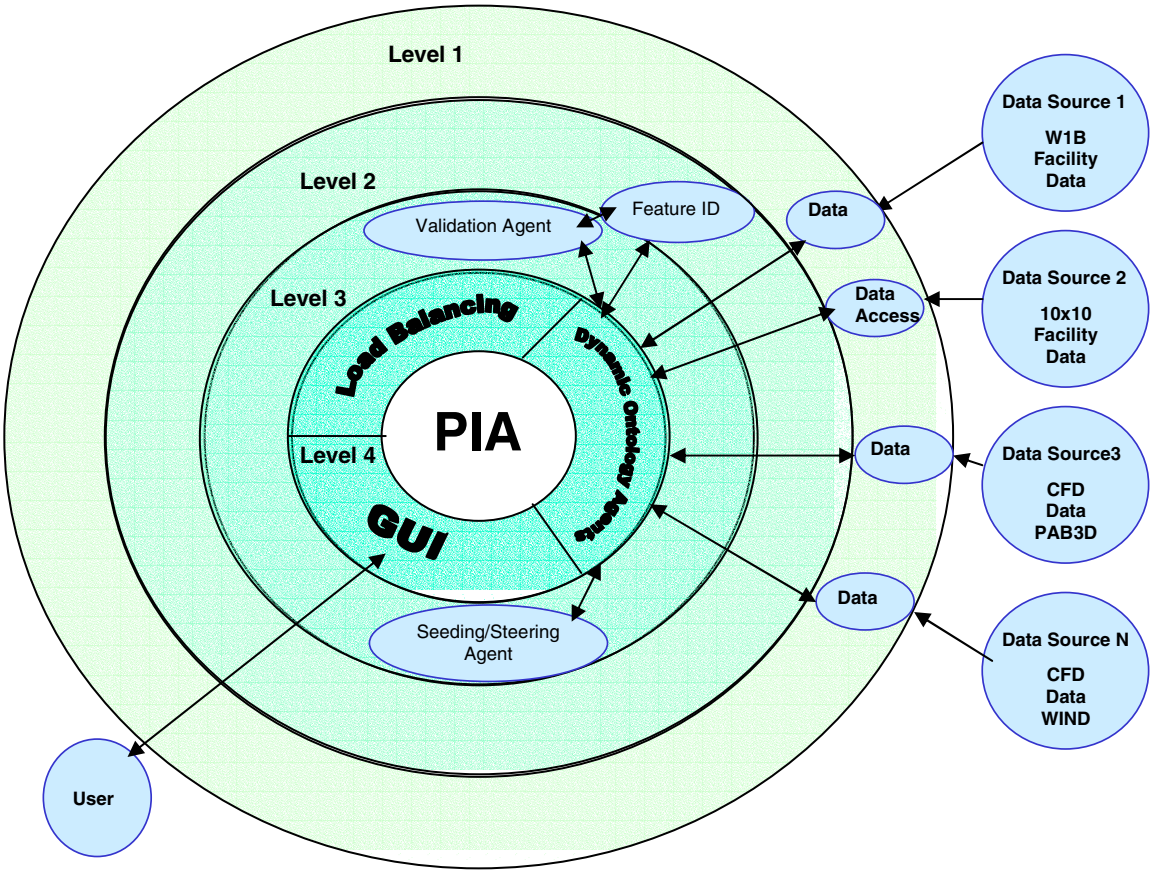


Figure 8.—Employing Intelligent Agent Technology around PIA

This can require (as has been done in demonstration efforts) not just a simple transport of information, but a recombination and synthesis of several information sources into a form needed by that wrapped application.

A more detailed view of a data discovery system is shown in figure 8. PIA is the core of the system, surrounded by agents that provide data fusion services. Since PIA is itself distributed, we refer to a PIA "collective", which is a set of PIA servers told to trust one another. The "rings" in the figure represent the level in the generalized JDL model as discussed above. The innermost ring surrounding PIA provides basic services that support level four, assessment and control of the data fusion process itself. We include ontology services, graphical user interfaces to monitor and control the processes, and load balancing of computational resources. These are just some of the possibilities, not intended to be a complete list. Data from various sources are acquired via Data Access Agents that communicate with PIA and other agents via the ontology services and an agent communication language. The Data Access Agents also perform level-one fusion operations, such as data alignment and object identification. The other agents deal with objects at different levels of the fusion model. For example, the Feature Identification (ID) agent combines objects and interprets the combined entity within the context of an application. The validation agent takes experimental and computational objects processed at other levels and evaluates how well they compare.

While the entire data discovery system in figure 8 is a long-term objective, much can be learned through applying PIA to current experimental and computational processes.

## VIII. Conclusion

As described above, PIA is an environment where data can be generated collected, archived, and accessed for analysis and complex problem solving. The PIA environment can be used as the basis for a discovery system that will enable technologists to solve increasingly complex interdisciplinary problems in future data-intensive environments. Technological processes can be documented and archived for future use by third-parties thus enabling them to devise derivative works based on the original process. PIA can be distributed over many computer resources

enabling efficient use of resources while gaining access to many data archives, simulations and data analyses. Intelligent queries can be formed to discover solutions never considered before.

This data discovery environment will be able to:

1) Integrate Data (simulations, experiments)
2) Integrate Models (CAD Models, VRML files)
3) Integrate Applications (mesh generators, numerical simulation, visualization, data standards)
4) Integrate Processes (product life cycle management, knowledge management)
5) Integrate People (collaborative environments)

By integrating the above, the PIA environment can achieve all the functionality required for a data discovery system.

## References


[1]James Douglas Stegeman, Richard A. Blech, Theresa Louise Benyo, and William Henry Jones, "Integrated CFD and Experiments (ICE): Project Summary," NASA/TM—2001-210610, NASA Glenn Research Center, Cleveland, Ohio, December 2001.

[2]The American Society of Mechanical Engineers. "Integrated CFD and Experiments RealTime Data Acquisition Development," ASME 93–GT–97, 345 E. 47th St., New York, NY 10017, May 1993. Presented at the International Gas Turbine and Aeroengine Congress and Exposition; Cincinnati, Ohio.

[3]William Henry Jones. "Project Integration Architecture: Application Architecture." Draft paper available on central PIA web site (http://www.grc.nasa.gov/WWW/price000/pub/tm00.pdf), March 1999.

[4]Theresa L. Benyo. "Project Integration Architecture (PIA) and Computational Analysis Programming Interface (CAPRI) for Accessing Geometry Data from CAD Files," NASA/TM—2002-211358, NASA Glenn Research Center, Cleveland, Ohio, March 2002.

[5]Gorodetski, V., Karsayev, O., and Samoilov, V., "Multi-agent Data Fusion Systems: Design and Implementation Issues," Proceedings of the 5th International Conference on Information Fusion (FUSION-2002), Annapolis, MD, July 8–19, 2002.

[6]Reinoso Castillo, J., et al., "Information Extraction and Integration from Heterogeneous, Distributed, Autonomous Information Sources A Federated Ontology-Driven Query-Centric Approach," IEEE International Conference on Information Integration and Reuse. To appear. Available at http://www.cs.iastate.edu/~honavar/Papers/indusfinal.pdf

[7]Griss, M. L., and Pour, G., "Accelerating Development with Agent Components," Computer, May 2001.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. AGENCY USE ONLY (*Leave blank*) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | January 2005 | Technical Memorandum |

**4. TITLE AND SUBTITLE**

Exploring Diverse Data Sets and Developing New Theories and Ideas With Project Integration Architecture

**5. FUNDING NUMBERS**

WBS–22–617–91–40

**6. AUTHOR(S)**

Theresa L. Benyo and William H. Jones

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

National Aeronautics and Space Administration
John H. Glenn Research Center at Lewis Field
Cleveland, Ohio 44135–3191

**8. PERFORMING ORGANIZATION REPORT NUMBER**

E–14935

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

National Aeronautics and Space Administration
Washington, DC 20546–0001

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

NASA TM—2005-213415
AIAA–2004–6360

**11. SUPPLEMENTARY NOTES**

Prepared for the First Intelligent Systems Technical Conference sponsored by the American Institute of Aeronautics and Astronautics, Chicago, Illinois, September 20–22, 2004. Responsible person, Theresa L. Benyo, organization code RTS, 216–433–8723.

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**

Unclassified - Unlimited
Subject Category: 61 Distribution: Nonstandard

Available electronically at http://gltrs.grc.nasa.gov

This publication is available from the NASA Center for AeroSpace Information, 301–621–0390.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** ***(Maximum 200 words)***

The development of new ideas is the essence of scientific research. This is frequently done by developing models of physical processes and comparing model predictions with results from experiments. With models becoming ever more complex and data acquisition systems becoming more powerful, the researcher is burdened with 'wading through' data ranging in volume up to a level of many terabytes and beyond. These data often come from multiple, heterogeneous sources and usually the methods for searching through it are at or near the manual level. In addition, current documentation methods are generally limited to researchers' pen-and-paper style notebooks. Researchers may want to form constraint-based queries on a body of existing knowledge that is, itself, distributed over many different machines and environments and from the results of such queries then spawn additional queries, simulations, and data analyses in order to discover new insights into the problem being investigated. Currently, researchers are restricted to working within the boundaries of tools that are inefficient at probing current and legacy data to extend the knowledge of the problem at hand and reveal innovative and efficient solutions. A framework called the Project Integration Architecture is discussed that can address these desired functionalities.

**14. SUBJECT TERMS**

Expert systems; Decision support systems; Management information systems; Knowledge management; Systems integration

**15. NUMBER OF PAGES**

21

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | |