ECS INDEPENDENT ARCHITECTURE STUDIES

EVALUATION PROCESS

In performing our analysis of the independent architecture studies, we have extracted key recommendations made by the teams and have summarized the potential impact of these recommendations on the ECS Program. We found that the clearest way to characterize these recommendations is to label them according the following five major groups:

a) ECS - Items in this category are currently under full or partial consideration by the HAIS team, or are currently incorporated into the HAIS architecture. The detailed approach or implementation used in the HAIS architecture may vary from the approach suggested in the university study, however the primary functionality remains comparable.

b) PDR (Preliminary Design Review) - Items in this category are of immediate relevance to the PDR and should be assessed as implementation solutions at the PDR. Whether or not the item is currently under consideration by HAIS, the particulars of the suggested implementation warrant specific attention from the HAIS team.

c) EVO (Evolvability) - Items in this category should be considered for future generations of the ECS architecture. These are items that the current architecture must eventually be able to support through a reasonable evolutionary path. We have further divided this category into both short-term (S) and long-term (L) classifications. The ECS architecture must be able to support short-term items within two to three years, and it must be able to support long-term items within three to five years. The long-term items are considered insufficiently mature, and need to be re-assessed at a later time. In some cases, items in this category are marked both short and long term. This is done for suggestions that have elements that fall into both classifications.

d) R&D (Research) - Items in this category are considered worth investigation, but their potential cannot yet be realized without further research. This category is applied to unproven technologies and promising concepts. Further research in these areas is likely to provide benefit to the ECS program.

e) PRG (Programmatic) - Items in this category may require further review at the programmatic level. In most cases, they are recommendations which are outside the scope of the current ECS program, or they would require re-organization of the program structure in order to be implemented.

In many instances, individual suggestions from the university teams were placed in more than one of the above groups. This is because these groups are not mutually exclusive. For example, a single suggestion may be seen both as needing assessment for the PDR while a similar concept might already be under consideration by the ECS team. Likewise, a concept may be seen as important for the future evolvability of the system while it also has certain elements that require further research.

Summary and Analysis of the UCB Independent Architecture Study

The U.C. Berkeley team presents a "database centric" arcchitecture for EOSDIS. Based on the analysis of several user scenarios, the team concludes that a completely flexible approach to the "standard production" vs. “compute-on-demand” strategy is required for all processing within EOSDIS. This approach is necessitated by the evolutionary nature of the production algorithms, the desire of high-end userss to have products tailored to their individual research needs, and the unpredictable nature of product demands.

To support this strategy, the team presents a “database centric” architecture that advocates support for advances in Object-Oriented Relational DataBase (OORDB) technology which will form a substantial portion of the proposed software architecture for the Science Data Processing Segment. SQL-* (current vendor implementations of abstract data types and distributed query language) is proposed as the standard query language that will be used to support the proposed middleware. Location and data distribution transparencies will be supported by the middleware. Database triggers and views are employed to support a dynamic "eager vs. lazy" evaluation production strategy that can respond to changing needs on a product-by-product basis. The team also proposes an incremental implementation of a rich Earth Science type library using abstract data type mechanisms. An alliance with one or more commercial database vendors and HSM vendors is proposed as the mechanism to deliver a tertiary-storage-backed distributed database engine.

The team proposes a system design based on a "2 + N" model that reallocates the functionality of EOSDIS across 2 superDAACs, and N peerDAACs, with the superDAACS responsible for processing and archiving level 1 and (most of) level 2 data, and the peerDAACs responsible for generating and distributing the remainder. The team suggests that true archival of the data only be performed for level 1, with the remainder of the data being reproducible on demand. The architecture employs a distributed approach to both computing and storage, based on the team's analysis of technology trends and price-performance curves that favor large numbers of distributed centers employing commodity technologies over a smaller number of centers employing higher end technologies.

The team identifies potential risks associated with the proposed software and hardware architecture, and develops a cost analysis based on the proposed functionality and their assessment of technology trends. The team also includes an initial design of a representative database schema for the scenarios presented, and illustrates the use of queries in accessing and manipulating the data.

In the following sections, we identify and analyze the potential impact of the team’s major ideas and suggestions to the ECS program using the previously described evaluation classification. The team technical contribution is categorized into three areas: information management, network management, and other architecture issues.

Information Management

- **Rearchitect DAACs and SCFs**

  Categorization: PRG

  The UCB team suggests redistributing the current DAAC and SCF functionality in a “2 superDAACs + N peerDAACs” model. In addition to storing the entire collection of raw data, the superDAACs will precompute the results to commonly requested queries and perform processing required for some ad hoc queries. The additional N peerDAACs will respond to ad hoc queries and store data sets commonly requested by a particular subset of the Earth Science community. The peerDAACs will be fashioned from commodity computing and storage technologies and have no formal performance and/or reliability requirements, making them smaller and cheaper than the superDAACs.

The redistribution of functionality as suggested has severe programmatic restructuring implications for EOSDIS. However, it should be noted that the current ECS architecture does not depend on any particular configuration of "DAACs" and "SCFs". Rather, it allows an arbitrary number and configuration of "providers", with the actual roles dictated by policy. Support for service reconfiguration and migration is one of the features integrated into the architecture as a result of the System Requirements Review and as demonstrated at the System Design Review.

- **Focus on ad hoc queries and flexible strategy for computing standard products**

  Categorization: PDR, EVO (S/L), PRG

  The premise of the UCB team is that there will be few true "standard" products over the course of time. Rather, they believe that users will gravitate towards a more dynamic interaction model, in which desired data is discovered, produced and requested through an ad hoc querying process, as opposed to more structured requests for standard products. Their proposed architecture reflects this belief, providing all processing (both standard products and ad hoc requests) through a common mechanism. This mechanism allows optimization between precomputed "standard" products, and compute-on-demand "ad hoc" requests at all levels of processing, responding dynamically to user requests and the availability of computational/storage resources.

  Support for a dynamic compute vs. store tradeoff on a per product basis has been integrated into the system design, again based on SRR feedback. Current consensus among the instrument teams, and the results of the Ad Hoc Working Group on Production, appear to predict a somewhat less dynamic environment than that indicated by the Berkeley team. This environment reflects a fairly tight interdependence of products and some longer term algorithmic stability (after initial fluctuations), leading to more "standard" products than assumed by the UCB team.

  Additionally, many of the instrument team algorithms are based on legacy instrument algorithms. The re-architecting of these algorithms to support a database-centric environment (where inputs are obtained from, and outputs generated into the database) could provide additional flexibility in supporting lazy vs. eager evaluation. However, this re-architecting comes at some cost, and it is unclear to what extent dynamic lazy vs. eager evaluation can be used in the highly interdependent processing scenarios.

  The current production architecture could support a scenario more biased towards lazy evaluation by reallocating computational resources to ad hoc queries. The details of these features in supporting a dynamic lazy vs. eager evaluation optimization will be addressed at PDR in consultation with the Ad Hoc Working Group on Production. Longer term technology evolution in resource management and dynamic scheduling should also be tracked.

- **Different processing modes**

Two major processing modes are proposed by the ~~S2K~~ UCB team.

- **Periodic Cycling of the store ("queries from hell")**

  Categorization: PDR, EVO (S/L)

  These queries typically require access to large portions of the storage repository, e.g. to "reprocess" an entire dataset using a revised algorithm. Such queries may run for days, or even weeks. The UCB team suggests "batching" such queries periodically, so that a single sequential pass of the stores may support a number of such requests. Techniques are required to recognize this type of query, and the middleware must support collecting together, executing, and disassembling the appropriate answer for such batches.

  This was generally received as an excellent suggestion, and is applicable to the current ECS design as well as the UCB architecture. It should be addressed by the ECS team at PDR. Longer term technology evolution should also be employed to support better a priori identification of resource intensive queries, and to provide effective resource scheduling over the extent of potentially long duration queries.

- **Eager vs. lazy evaluation**

  Categorization: ECS, PDR, EVO (L), R&D, PRG

  The output of every node can be evaluated eagerly when inputs become available; or it can be evaluated lazily when its output is needed. The team believes that all "products" within EOSDIS need to be considered for this optimization, allowing dynamic distribution of resources to the push (eager) and a pull (lazy) sides of the system as motivated by user demand. The optimization cost model and algorithm to support such a dynamic resource allocation problem is very complex, and further research is needed. In addition, queuing and workflow management techniques for organizing and scheduling triggers and views require plans for long term evolvability and R&D effort. A short description of the cost model and an example of using database triggers and views to support this evaluation are provided in section 3 of the report.

  This item is closely related to the discussion "Focus on ad hoc queries and flexible strategy ..." above. The current ECS architecture supports both precompute-and-store, and compute-on-demand generation of ECS "products", as recommended by the science community. There is additional interest in determining how flexible the PDR implementation will be in this regard, and how effective the architecture is in reallocating computational resources in such a scenario. At a programmatic level, there will be a need to establish policy on the lazy vs. eager evaluation question. This may require streamlined decision processes for migrating algorithms between these classes of execution.

• **SQL-* is the principal access language**

Categorization: ECS, EVO (S/L)

The UCB team suggests the use of SQL-* as the query language to retrieve data, and as the intermediate language for communicating among middleware, target DBMSs, and clients. Appropriate translation to SQL-* is needed to support a variety of client languages and target DBMS languages. And by having SQL-* as the intermediate language, the required number of translations among foreign underlying data servers and clients are reduced from n-square to 2n. The UCB team suggests that database vendors employing SQL-* will be developing a number of such gateways over time in order to integrate legacy and new systems. The features of SQL-* required to support the proposed architecture are described in the sub-bullets below.

The current ECS architecture plans to support a number of protocols, including SQL dialects with type extensions. Consistent with the philosophy of heterogeneity and custom client interfaces, the ECS architecture plans to support multiple protocols, such as HTTP and Z39.50. This leaves service implementation decisions up to the provider, allowing protocols well matched to the provider's capabilities and desired level of service. Because of the volatile nature of most of these standards, the ECS team has been hesitant to "bank" on just one. There are particular concerns with SQL-* in that it represents current vendor implementations of portions of the current SQL3 standard. With the SQL3 feature set unclear, and 3-5 years away from standardization, the ECS team has been hesitant to commit to what is, in essence, an unknown standard. Given the shorter term deliverables (release A in 12/96), reliance on SQL-* and SQL3 alone seems risky. ECS provides an architecture that allows multiple protocols to be supported since there are heterogeneous Earth data systems. This architecture doesn't exclude the support for SQL-* protocol. Protocol trade studies are underway at the present time and will be reported on at PDR. Longer term, the ECS team will need to follow the SQL3 and OQL (OMG) standards, and will need to support these standards as commercial implementations are realized.

- **Type extension & functions & type library**

  Categorization: ECS, EVO (L), R&D

  SQL-* provides the mechanisms required for type extension, but it does not implement the type libraries that will be required for ECS. While a number of more common types will be provided by commercial database vendors (arrays, multimedia extensions, etc.), it is not clear to what extent ECS-specific data types (e.g. spatial types, time series, event models, etc.) will be implemented. The team suggests that ECS should define the SQL-* type libraries that will extend SQL with EOSDIS-specific data types and functions. The team believes substantial effort is required in this area, since standard libraries for scientific data such as those provided within NetCDF and HDF will need rework to provide the right level of abstraction for a SQL-* representation.

  The ECS data model is being developed through Data modeling and Data server design efforts. The model includes Earth Science Data Types (ESDTs), which provide the visible types and services exported by the Data Server, as well as Computer Science Data Types (CSDTs) that are used as implementation vehicles for the ESDTs. This effort provides much of the "functionality" called for by the UCB team, but without the direct linkage back to an extended SQL dialect. The ECS team needs to track the extended relational and OODB implementations to insure compatibility of it's type library with eventual SQL dialects, allowing development of long term functionality in a database-centric environment.

- **Triggers & views for production management**

  Categorization: EVO (L), R&D

The UCB team suggests database *triggers* and *views* as the appropriate mechanisms to support eager and lazy evaluation. Triggers are used to support eager evaluation, allowing arbitrary processing sequences to be initiated in response to events(e.g. receipt of data). Views contain descriptive information pertaining to the underlying data sets, and can be used for lazy evaluation. More R&D efforts are required to advance the concept of triggers and views to the point where they can realistically be used for production management. This includes work on data consistency support for view update, transactional semantics, cross-site triggers, resource management, and process scheduling. Discussions of the usage of triggers, their limitations, and possible extensions for production management are discussed in section 3.4.2 of the report.

The current ECS approach uses a production management strategy based on graph theoretic algorithms (dependency graphs), resource management strategies, and current process queuing and process management techniques. The UCB team's approach suggests use of trigger and view extensions within the database to accomplish these features. When coupled with processing methods that reside in the database, this would provide a production system that was tightly coupled with the database-based methods, affording the flexibility in lazy vs. eager evaluation the team believes is required. These extensions, particularly in extending transactional semantics of triggers in cross-site execution, and in resource management and process scheduling, are largely research problems at the present time. Hence, ECS should monitor the progress of commercial R&D efforts and preserve the ability to incorporate new technological capabilities over the long term.

- **Access methods & index techniques**

  Categorization: ECS, EVO (L), R&D

  The ability to add efficient spatial data retrieval, content-based retrieval, and temporal data access requires the ability to specify complex indexing techniques along with data type extensions. Research is needed for advanced index structuring to efficiently support the retrieval of various spatial data types within ECS.

  The ESDTs and CSDTs of the ECS architecture described above must take into consideration appropriate indexing schemes for spatial or value-based (content-based) retrieval. These schemes need to be provided as part of the data type implementations described above. While a number of common spatial indexing schemes exist, other schemes required by ECS may be esoteric, and will require longer term R&D efforts. The ability to support such schemes in the long term needs to be demonstrated as evolutionary criteria.

• **Middleware software**

The UCB team proposes a layer of software that sits between the client and the target DBMSs to support the following features:

- **Location transparency without DCE & CORBA**

  Categorization: ECS, EVO (L)

  The UCB team proposes implementation of location and data distribution transparency through the middleware. This middleware provides automatic mapping between a users' location-independent request, and the location-dependent data sources through a distributed query decomposition. The mapping and decomposition are entirely transparent to the user.

The ECS architecture proposes service location transparency through DCE RPC, directory service, and trader functionality, with migration to an OMG base in the longer term. Because the ECS architecture provides service interfaces across a number of protocols, it uses a more general request / reply mechanism on top of a DCE/CORBA foundation. This basic mechanism can be used to "carry" a number higher level "protocols", including extended SQL dialects, HTTP, RPC (DCE IDL), and object messaging (OMG IDL). This is a fundamentally different approach than that taken by the UCB team, but one which provides for location transparent service invocation across a network of heterogeneous providers, a scenario that seems likely in the GCDIS scenario that ECS aims to support. The UCB solution suggests developing a common language (SQL-*) in which all providers agree to converse. The ECS team will need to illustrate adaptability among these various scenarios as a long term evolvability test.

- **Distributed query optimizer for parallelism, load balancing, and copy selection**

  Categorization: ECS, EVO (S/L), R&D

  A distributed query optimizer is needed to automatically decompose queries and to select an optimum access plan that will reduce the total execution cost of a query. Based on the data distribution (replicated and fragmented components), the processing load of each site, the network cost, and the complexity of the SQL-* query, the optimizer will attempt to exploit the available parallelism among a subset of the 2+N sites. These issues, as well as that of overall resource management (load balancing) are among of the more challenging research topics being pursued by the database research community.

  This is one of the more challenging research efforts required for transparent distributed site search. The ECS architecture supports evolution to a system capable of exploiting advances in these research areas, while minimizing the impact of delays in such technologies. The DIM / LIM hierarchy within ECS will employ distributed query optimization techniques, parallel query execution, copy selection, and load balancing when they are available. However, it is not completely reliant on these advances for successful operation. A number of techniques can be employed within the DIM to provide distinct global "views" of the data to various populations within the ECS community. As distributed query optimization matures, these views can migrate from static representations to more dynamic ones.

• **Should adopt a “DBMS-centric” view point**

The UCB team recommends using an advanced object-relational DBMS as a local storage and processing engine.

- **Instantiate all functions in a DBMS**

  Categorization: EVO (L), R&D

  All user-defined functions and user-defined access method should be stored and executed within DBMS boundaries. Maintenance and portability of these functions are still R&D topics.

This approach has a number of advantages, including the ability to simply migrate algorithms from eager to lazy evaluation, and vice-versa. Because all required production functions are housed in the database, they can be invoked in a uniform manner. This approach also provides for simple execution of alternative processing scenarios, as various implementations of data type "methods" can be invoked simply through the SQL-* interface. This approach also allows lineage information to be preserved through simple database operations and system audit trails. However, such an approach requires all algorithms to be developed in a database-centric model, with inputs being "SELECT"ed from the database, and results being "INSERT"ed into the database. For new algorithms and appropriately trained instrument team programmers, this may not be an issue. However, for the substantial legacy instrument code investment, and more traditional "file in - process - fileout" programming models, such an approach requires additional support. The ECS team needs to consider the ramifications of migrating functionality into a database-centric model downstream, as part of its evolution assessment.

- **No file systems besides the one use for DBMSs**

  Categorization: EVO (L), R&D

  This is consistent with the UCB team's DBMS-centric approach, and is a laudable goal -- providing a single storage abstraction through the database, and handling all "reads" and "writes" to that storage abstraction through database mechanisms. In order to support legacy systems, a software "wrapper" could, in principle, be used to interface file-based algorithms with ECS. However, this type of encapsulation process is very time consuming and has severe limitations depending upon the code structure of the legacy systems. Moreover, ECS performance will degrade significantly.

  Longer term, we believe computer system evolution to move in this direction. We are starting to see "database" features in existing operating systems (Macintosh Finder file properties and search, Microsoft's next generation operating systems, etc.). This will tend to blur the distinction between file systems and DBMSs, a trend that ECS will need to plan for in its evolutionary path.

• **Integration of hierarchical storage management (HSM) and DBMS**

Categorization: EVO (S/L), R&D

The UCB team suggests direct integration of the HSM with the database engine. This provides the required control over caching mechanisms that may not be available through a cached file interface between the DBMS and a file-based HSM. The team suggests substantial investment in advancing the state of this technology, based on its experiences as outlined in section 3.3 of the report.

The overriding message from the UCB team here is to plan for DBMS systems that are integrated with tertiary storage systems, rather than laid on top of file abstractions on those systems. To that end, they suggest working closely with database and HSM vendors to develop these integrated products. This approach seems advisable even in the more heterogeneous design of ECS, as it will allow ECS to leverage commercial advances in this area. This will provide more efficient and cost effective implementations of tertiary-storage-backed database systems from an ECS perspective, regardless of whether this is the only storage mechanism supported by the architecture.

- **Incrementally define Earth Science type library**

  Categorization: ECS, PDR, EVO (S/L)

  Substantial extension to S2K, NetCDF and HDF metadata is required to be useful in EOSDIS. Development of the earth science type library is a major effort, and one that will require periodic refinement to insure adequate support for science applications over the life of ECS. The current development effort includes definition of a core metadata model and design of a rich Earth Science Data Type (ESDT) taxonomy with associated services / methods. The ECS team should review this design in some detail at PDR, and should continue to evolve the types and methods as time goes forth.

- **Incrementally define SQL-* schema and data dictionary**

  Categorization: ECS, PDR, EVO (S/L), R&D

  A data dictionary of common terms and synonyms to be used by EOSDIS scientists, and textual descriptions of data elements need to be defined. Moreover, an SQL-* schema needs to be defined in an iterative and evolutionary manner to represent EOSDIS data and to support the middleware. Techniques and tools to semi-automatically build and integrate these schema and the data dictionary are currently not well understood, even by the research community.

  Again, this is being done within ECS in the data modeling (core metadata definition), Data Server design (schema definitions and extensions), and Data Management design (data dictionary) efforts. These designs should be reviewed at PDR. Additionally, the design needs to demonstrate the ability to adapt to schema and data dictionary changes over time -- as new products are added, new attributes used to describe existing products, and new or modified terminology added to the data dictionary. Longer term R&D efforts should include development of tools for schema analysis, integration, and federation. This will support management of the federated / integrated schema hierarchy in a dynamic environment. Tools such as these are one of the targets for prototyping.

- **Client-middleware & middleware-server protocols (Gateways)**

  Categorization: ECS, PDR, EVO (S/L)

  Such protocols will allow the inter-communication among Client applications, Middleware, and Database servers. SQL-* will be used as the intermediate language. The team discuss the interoperability and portability of two protocols: API and FAP in section 3.5.1. Since SQL-* is proposed as the query language, a combination of APIs and gateways is needed to achieve vendor independence (portability and interoperability).

  The team contrasts the two protocols with other client-server protocols, including Z39.50, CORBA, and HTTP (see section 3.5.2). In summary, the protocols are very important to the success of building a large distributed information system that includes both legacy and new data servers. Current standards and gateway technology to facilitate interoperability among a number of different servers are still immature. Therefore, a close look at the protocol selection and standard is strongly recommended.

The current ECS architecture provides "gateway" functionality at the Data Server interface. The Data Server exports a list of protocols and services that it will respond to, and clients attach to those services using the appropriate protocols. The Data Server may convert the accepted protocol in any way it desires internally. This approach supports heterogeneous providers by allowing them to "advertise" services in a native protocol, rather than putting the burden on them to convert to a "standard" protocol. The ECS team's requirement to develop components for GCDIS reuse and to prepare for a heterogeneous environment has led them to this design point. The details of this design will be discussed at PDR. Extended type queries in the flavor of SQL-* or SQL3 will be an important part of this protocol suite, but they will most likely not be the only query and delivery protocol. The rapid acceptance and continued development of HTTP is likely to continue, and it is reasonable to believe that "providers" in the longer term will want to provide services via this and other protocols. Nevertheless, the ECS team needs to consider evolution of the system to a predominantly SQL-* / SQL3 based infrastructure, and show how the system can adapt to such a scenario.

- **Standards evaluation (SAIF, OGIS, OMG, SQL/MM)**

  Categorization: PDR, EVO (S)

  ECS should make a major effort to influence these standard activities so they become even more useful over time. The ECS team needs to show at PDR it's involvement in these and similar standards efforts, and demonstrate infusion of appropriate components into the core metadata and Data Server designs. Because standardization is a longer term prospect in a number of these cases, the ECS design needs to be able to react to changes in the status of these standards -- e.g., how will widespread adoption of OGIS affect the core metadata model and Data Server designs.

- **Don’t use CORBA**

  Categorization: ECS, PDR, EVO (S)

  CORBA (Common Object Request Broker Architecture) is a client-server request-reply protocol defined by the Object Management Group (OMG). It is a standard way for a client to invoke a server. The team enumerates several reasons why CORBA should not be used as currently defined, see section 3.5.2. ECS needs to monitor this standard closely.

  The approach to CORBA adoption is outlined in the ECS System Design Specification, and will be addressed again at PDR. The current approach employs DCE in the release A/B timeframe, with migration to CORBA in the release C/D timeframe as the technology matures. The "Don't use CORBA" argument is addressed at the fundamental search mechanism supported in the system. The UCB team suggests that a CORBA foundation will lock the ECS design out of commercial advances in cross-site, distributed search, and force development of similar functionality on a CORBA foundation. ECS needs to address this concern by showing how the architecture can evolve to support the commercial distributed search capabilities that will evolve on a SQL-* / SQL3 foundation.

- **Data lineage support**

Categorization: ECS, EVO (S/L)

Heritage of the data need to be captured and manipulated using a DBMS.

The current ECS architecture supports capturing of data lineage information. This will most likely be done in a DBMS environment, as suggested by the UCB team. Within ECS, lineage will be explicitly tracked through production "logging" to the appropriate information repository. The UCB team points out the benefits of database technology in supporting automatic lineage capture through mechanisms such as DBMS triggers and audit trails. Use of such mechanisms should be addressed in discussions of system evolvability, ensuring that the overall system can take advantage of these capabilities within various Data Server implementation strategies.

Network Management

- **Use DCE**

  Categorization: ECS, PDR

  DCE (Distributed Computing Environment) is network operating system software that supports a client/server model across platforms. The software is currently supported by many vendors.

  The use of DCE is recommended for lower level network services and network management. Given the SQL-* approach, this will require SQL-* middleware vendors to build distributed query processing and schema integration/analysis capabilities on a DCE-based infrastructure to provide network security and inter-site/process communication for heterogeneous platforms. Presumably, DCE/RPC is not proposed as the query and response delivery mechanism, as it has a similar request/reply model to that of CORBA. This will require a DCE-based approach to the SQL-* middleware layer to implement the approach proposed by the team.. The ECS team has adopted DCE as both the network delivery (request/reply) and network management layers for release A/B, and should be prepared to address the implementation approach at PDR.

- **Adopt System Network Management (SNMP)**

  Categorization: ECS, PDR, EVO (S/L)

  SNMP is a set of tools and standards that allows the systems to scale to thousands of nodes and still be manageable. Netview product suite is identified as one of the tools.

  ECS has standardized on SNMP. The design details should be reviewed at PDR, as should plans for taking advantage of evolution in network management technology.

Other Architectural Issues

- **Should do "just-in-time" hardware acquisition (6 months)**

  Categorization: PDR

  The report presents the team's assessment of technology trends in CPU, disk, tape, and network technology. The team bases its cost models on "just in time" hardware acquisitions (6 months prior to operational use) to reduce hardware costs.

  Current ECS acquisition is phased as suggested by the UCB team. However, much of the ECS hardware architecture is based on "high end" components (FDDI LANs, HiPPI, shared disk, etc.), which may have longer procurement cycles than commodity technologies. The ECS team acquisition strategy needs to be reviewed at PDR. This should cover the current acquisition strategy and basic assumptions, including the phased capacity planning (i.e. production demand basis), and procurement constraints.

- **Use automated techniques to reduce user support**

  Categorization: ECS, PDR, EVO (S)

  The team proposes to move from a "human-intensive" support system to an "electronic-intensive" support system using on-line help, Mosaic and the World Wide Web (WWW).

  The current ECS plan is to utilize electronic support mechanisms to the extent possible. This includes on-line help facilities accessed through standard interfaces such as WWW browsers. These plans should be reviewed at PDR. Evolvability to take advantage of advances in these areas should also be addressed.

  - **share DBAs among superDAAC and peerDAACs**

  Categorization: PRG

  The UCB team recommends a review of staff assigned to DBA-like activities, suggesting that a single individual can support a large number (on the order of 25) of smaller facilities. It is not clear how this recommendation relates to the current ECS approach, which has DBAs concentrated at the DAACs. Programmatic review of this distribution is suggested.

  - **Email support, electronic documentation ....**

  Categorization: ECS

  Electronic documentation, help and email support are all planned within the current ECS approach. The ECS team should continue to look for ways to efficiently support the user base, using electronic means to leverage human effort.

- **Technology & user scenarios assessment**

  Categorization: ECS, PDR

Several valuable user scenarios (section 2 & appendix 3) and technology forecasts (Appendix 2) are described in detail in the report. The database schema design and SQL-* queries presented in support of the user scenarios are described in Appendix 4. The team also presents a risk assessment of reliance on database technology in section 1.10. We highly recommend that the ECS team take a close look at these studies. Implications of these scenarios for the current architecture should be addressed at PDR.

- **Cost assessment (advanced technologies, rearchitect DAACs, ...)**

  Categorization: PRG

  Section 5 presents a 6-year budget of the cost to procure and operate the main components of the proposed architecture based on the prediction of advanced technologies in software/hardware, and on the redistribution of functionality among the "2 + N" DAAC model.

  We recommend a programmatic review of some of the approaches suggested by the UCB team. In particular, consideration should be given to streamlining the procurement process and hence supporting "just in time" acquisition. Additionally, the program may want to look at ways to distribute functionality and move into more commodity technologies as suggested by the peerDAAC concept. We realize that this could have significant programmatic considerations, and may not be achievable in the short term.

- **Influence small to mid-size COTS vendors & minimize in-house development**

  Categorization: PRG

  EOSDIS should focus on working with COTS software rather than develop large proprietary systems. The team recommends that ECS should work with at least two SQL-*/HSM database vendors and should purchase COTS system management tools based on the SNMP protocol.

  This appears to be an effective approach to developing some of the larger system components that are well aligned with commercial product interests. The program should find ways to accelerate development through outsourcing where program technology interests are suitably aligned with commercial interests. This is particularly true in the marriage of database and HSM technologies, and in the development of distributed query capabilities.

- **ESN between superDAACs and Internet for peerDAACs (bandwidth & cost ....)**

  Categorization: PRG

  The proposed architecture relies on the rapid improvement in network throughput and reliability. This recommendation is consistent with the current approach of providing ESN interconnection between DAACs, some high speed links to high volume QA facilities (SCFs), and Internet connectivity to the community at large. The ramifications of providing high speed ESN connectivity to a smaller number of DAACs are a programmatic issue.