

Principles for College Wide Data Management Infrastructure (Draft)

Goals for Our Centralized Data Infrastructure

1. Supported by a true college wide business need.
2. Improve data management and reporting, streamline operations, and get services out to departments.
3. Provide clean valid and up to date data.
4. Provide an infrastructure that allows comparison across years.
5. Provide keys in the data that allow us to join disparate sets of data easily. (This is crucial for complex reports, for instance faculty course evaluations with their course load and research expenditures.)
6. Provide departments with easy access to appropriate data. In particular everyone who submits data should be able to see and validate that data immediately. Ideally they should also be able to see the reports based on that data immediately.
7. Control access to data appropriately. Our security model must be very flexible. It cannot be based simply on departments; it must allow cross-disciplinary units access to data for courses and research done in their program, even though the faculty will be in other departments or colleges.
8. Provide central systems that are rich enough that departments and other units don't have to create duplicate and shadow systems.
9. In those cases where departments want to add functionality to existing systems (a legitimate use of shadow systems), allow them to integrate our data easily into their systems.
10. Provide connectivity between systems so we avoid multiple entry of data.
11. Use our choice of what to measure as a way of communicating strategy (ie: Balanced Score card and Key Performance Indicators).
12. Provide a comparative data for relative benchmarking. For example, when we provide detail data to departments we want to give them something to compare it too, perhaps rolled up data for other departments.
13. Eventually we may want to get comparative data from other units at Cornell and other engineering colleges as well (We will want to gather data on their practices as well so we can determine best practices.)
14. As we move to more collaborative relationships we need a more open data infrastructure to support work across disciplines and colleges. (Not only must it be more open, but it should also be more compatible or even unitary.)

Decision Criteria for Infrastructure Software Platform

1. Work with University Data Infrastructure.
2. Work with departments' data infrastructures (Access, Excel, Word, etc.)
3. Support a complex security model.
4. Allow us to “grow” or hire skilled developers.
5. Allow rapid development.
6. Have an active development community on campus, particularly among people doing similar work.
7. Software company looks to be viable over the long term. We don't want to settle on a tool that will not be in the mainstream in 5 years.
8. Not require too large a learning curve for existing staff. Ideally power users should be able to contribute tools to infrastructure.
9. Be easy to administer.

Goals for Data Systems

- A typical user in our office who unfamiliar with a given task or data set should be able to easily find the right file to work with. Once they find the right file, they should be able to open the file, find a desired set of records to view and/or edit. They should also be able to generate previously run reports easily.
- Power users should also be able to generate custom reports based on the base data, either within the tool or by exporting the data for further analysis in Excel or for a mail merge.

Disadvantages of Centralized Data Infrastructure, Things We Want to Minimize

- FTE cost in central office to maintain all of this.
- Computer Response time
- Training time

Data Management Tools

Below are the data management tools we are currently using or considering. I try to describe their strengths and weaknesses and in what situations they are best applied. You can characterize the tools by the functions they provide: Data store; data serving; user Interface rendering; custom client program creation, reporting; web page generation; and web page serving. FileMaker is the only one of the programs below that does all of these, generally the other tools do some of these functions better than FileMaker. FileMaker is not the most powerful in any of these categories but it is probably the easiest to learn.

Excel: My hope is that we will move away from using Excel as a data management tool. Excel is a very powerful tool for static data analysis but because its data management and analysis system is tied to cell location, it fails as a data management and reporting tool for large data sets or dynamic data. In some cases we may be able to use Excel's very nice reporting tools as a front end on a larger database such as FileMaker, SQL Server or Access.

FileMaker: A data base management and reporting tool. Can also be used to generate data driven web sites and custom programs. FileMaker and Access are the only desktop database tools that have been gaining market share. FileMaker is targeted at the individual user to large workgroup environment and has been rapidly developing its ability to work with Enterprise data systems. It is heavily used by Paul's counterpart in Arts, Cindy Sedlacek and Paul is very experienced in FileMaker.

Microsoft Access: A data base management and reporting tool. Integrates well with SQL server.

Microsoft Visual FoxPro: A data base management and reporting tool that can be used to create client programs. Fox has been losing market share but is a very powerful tool. Mike is a very experienced Fox programmer.

Microsoft SQL Server: A data server. Has no reporting capability and you have to use a different client program to input or manipulate the data.

Allaire Cold Fusion: Middle-ware that normally sits on the same server as the SQL server software and the IIS web server. It generates data driven web pages that are served by the web server to users viewing or entering the data from their web browsers. It allows for very rapid data driven web page creation and is very easy to learn to use. As a result student employees can be very productive in this environment.

Brio: A reporting tool that the campus has standardized on. A very powerful tool that provides all of the advantages of Excel's reporting tools to dynamic data.

Microsoft IIS Web Server: Serves web pages. Integrates well with Cold Fusion or the FileMaker web companion for serving data driven web pages.

Diagram of the tools roles <<to be created later>>

Existing infrastructure

Planned infrastructure

Decision Criteria

10. Work with University Data Infrastructure.
11. Support a complex security model.
12. Be able to grow or hire skilled developers.
13. Allow rapid development.
14. Have an active development community on campus particularly among people doing similar work.
15. Business looks to be viable over the long term. We don't want to settle on a tool that will not be in the mainstream in 5 years.

16. Not require too large a learning curve for existing staff (including developer).
17. Make good use of work study students.

Goals for Data Systems:

A typical user in our office who unfamiliar with a given task or data set should be able to easily find the right file to work with. Once they find the right file, they should be able to open the file, find a desired set of records to view and/or edit. They should also be able to generate previously run reports easily.

Power users should also be able to generate custom reports based on the base data, either within the tool or by exporting the data for further analysis in Excel or for a mail merge.

Programmer/analysts should be able to understand the existing data architecture well enough to expand it without having to re-engineer existing functionality or spend a lot of time deconstructing and documenting the existing data architecture.

Finally, should a truck hit any of us, our work should be well enough documented that someone else should be able to take it over without losing the work we've already completed. Ideally work in progress should also be able to be taken over by someone else as well, though in clearly this may not be feasible in many crunch projects. Future people in our positions should be able to tell what we did to get the result or report we did. In other words process documentation should be written and keep up to date and the data and files themselves should have some documentation about their source and any corrections made to them.

Implementing These Goals:

Obviously meeting these goals will require work at many levels. Office users will have to become proficient on the software tools we use. Standards for documentation and file location will have to be set and implemented. And a lot of documentation and re-factoring and conversion of existing data systems will have to be done.

Documentation

Documentation will have to be written at many different levels. In addition to the job description, each position will need to describe the processes they perform, the data and files they use or produce, and the relationships they manage. Paper file systems should be easy to relate to their electronic counterparts. Both file systems should be well enough documented that office staff can find the right file without having to look too hard. If a folder exists for a piece of paper or a file it should be obvious where to put it. Likewise it should be relatively easy to determine that a document doesn't exist either on the computer or on paper.

- Directory names should describe their contents.
- Every directory should have a "readme" file that describes what is in that directory and any related paper files.
- <<this list is incomplete>>

Documenting the Data (for Auditing and Other Purposes)

Much of the data I work with isn't accounting data that requires full audit trails, however it is often necessary to describe the source of the data and any transformations or corrections applied to the data. In the past this was not done explicitly and required the involvement of Cathy or others who knew the process used to gather the data. I would like to explicitly capture this data in the data set and documentation. The goal would be that any knowledgeable person could go into the data set and determine the how a particular value got into the data set. Typically this might be to answer a question from a department that's validating the data.

- Each record should have a source field to indicate where the data came from originally
- Each record should also have a corrections field to indicate corrections (and previous values) applied to data in the record. For a full audit trail, this would normally be implemented as a related file, but I think that's more complex than necessary.
- Each "original" field (field that's not derived from other fields) should be described in the data dictionary for the file. I may create a database to store the dictionary data for all the files I work with.

Documenting the Process to Gather, Transform, Validate and Report the Data

It is useful to know how and from whom data is gathered for each data set. This makes it easier to hand this process off to new staff and can help with validating and auditing the data. Associated with each data gathering process it would be useful to know:

- The source used for the data (the contact person or institutional data source)
- The process used to gather that data (email query, brio query, custom dbase for them to enter data, etc.)
- The process used to transform that data into the form needed for our dbases. This normally includes information about:
 - Key fields that need to be created
 - Ancillary dbases that need to be generated by hand (eg: a dbase that is derivative of the central data set such as a list of all the courses evaluated from the evaluation dbase.)
 - Cleaning operations (eg: making sure Faculty names are spelled correctly so the keys work to relate one file to another.)
 - Diagnostics run on the data to determine it's validity or the characteristics of the data (eg: how many records are not being used by the parent dbase.)
 - Validation steps that involve other people. (eg: Cathy reviewing the complete report to see if it makes sense.)
- Reports generated out of the data and when and to whom the reports are delivered

Documenting the Databases That Do This Work

The database documentation for databases is targeted at two distinct audiences: The database user, and power users and developers.

User Documentation

For the database user, there are two important aspects of the documentation, which files to use and what to do once you get there. With relational databases it is not always obvious which file to use to get a given report. The process documentation and directory readme files should solve this problem. I may also create a standard for a "start_here.fp5" database that contains all the documentation and buttons to generate all the reports for a given set of databases (eg: Faculty Data Summary or SIP/HR).

Once the user has started up the right dbase, the database will open into a layout that documents the reports and operations that are available from that database. They should be able to use that documentation to recreate any regular report previously or to navigate to the right layouts to enter new data. In general the database should be self-documenting. It may even be possible that the database will contain much of the data source and process documentation as well.

Power User/Developer Documentation

Documentation at this level should allow power users and developers to extend the database solution or extract data without recreating what already exists and without having to spend a lot of time figuring out what data is where. Much of the documentation listed above already addresses this issue but some additional information about the structure of the database is needed. This documentation will include things like Entity Relationship Diagrams and a "developers" layout or layouts that describe what special "tricks" I use to get the results I need. In addition to the ERD, I may include information about how the tables work together and why I run the reports in the tables I do.

Expected Basic Expertise of Staff Using Department Data

Data Entry

Data Use

Training Expectations

Standard Layouts, Scripts, and Field Naming Conventions for FileMaker

Note: Standards get developed based on actual need and practice and should be adjusted as experience warrants. As a result the following "standards" are targeted at FileMaker development. We will need to

expand them as we start using other programs. I anticipate doing this for Microsoft's SQL server and Allaire's Cold Fusion.

Data Access Control

Who should have access to what data is a very complex area to address and requires the involvement of many people. This document tries to frame that discussion.

In general, access control is structured into a hierarchy, where a person has access to the data of their subordinates. For instance the deans office of the college has access to the data of the departments, the department directors, chairs and managers to the data related to their faculty and staff and the faculty and staff to their own data. Historically the main exception was the data administrators who have access to all the data because they run the software that controls the data. Management structures are changing and no longer exist in a strict hierarchy. Faculty collaborations across departments or faculty teaching courses in other departments or faculty doing research administered in other departments are examples we deal with frequently. In these cases department managers may need access to data that they wouldn't have access to in a strict hierarchy. We need to have a security structure that allows for this flexibility from the outset, because adding special cases to a strictly hierarchical system can be very difficult. Managing hierarchical based security allows you to make assumptions such as each entity has one overseeing entity. This makes developing and administering the security system somewhat easier than the multiple oversight model I envision. There is also the possibility that in a collaborative project only one of the multiple departments that have oversight should be allowed to add records or change data. All of these issues need to be addressed.

It would be nice if the security architecture was similar in all the database solutions so I can leverage the development and administrative effort.

FileMaker version 5.5 provides a very flexible structure for administering security. <<More here about the architectural options, their advantages and disadvantages.>>

On aspect of the security system that I feel strongly about is that, the people who enter the data should have access to that data and reports specific to that data. (As a result I anticipate implementing a control structure that requires a unique log on for each user (or department?) and stores the creator ID with the record.)