Solving Business Problems with Good System Design

SYSTEMS SEMINAR CONSULTANTS, INC.

2997 Yarmouth Greenway Drive, Madison, WI 53711
Phone: (608) 278-9964 • Web: www.sys-seminar.com

Questions, Comments

- Technical Difficulties: Call 1-800-263-6317
- We have several hundred attendees, so we won't be able to answer questions live.
- Please email questions or comments to train@sys-seminar.com.
- A copy of the presentation and a recording of the webinar will be emailed out to attendees. This can be shared with people who did not attend.

Systems Seminar Consultants, Inc.

Consulting, Training, and Support

- SAS
- SQL
- ETL
- Reporting Systems
- Business Analytics
- Data modeling
- Automating Processes
- “Big Data”

Apply good IT and systems practices to real life business applications.

Work with Marketing, Finance, Retail, Insurance, Utilities, Manufacturing, Healthcare, Government organizations.

Presenter – Teresa Schudrowitz

- Senior Consultant, Project Manager, and SAS Trainer
- Data conversion, automated reporting systems, production support, application development, planning and analysis, and project planning.
- B.A. in Management Information Systems, B.A in Accounting, and M.S.in Management Information Systems.
- Presenter at at many local user group meetings, regional user group meetings, PharmaSUG, and SAS Global Forum.

Ad Hoc Vs. Production

- Ad hoc is less planned, put together quicker, and not as documented
- Should be flexible and well done
- We may use it for our next project
- Sometimes ad hoc becomes production

What is a Good System?

"It runs. I got my results. Everything is great!"

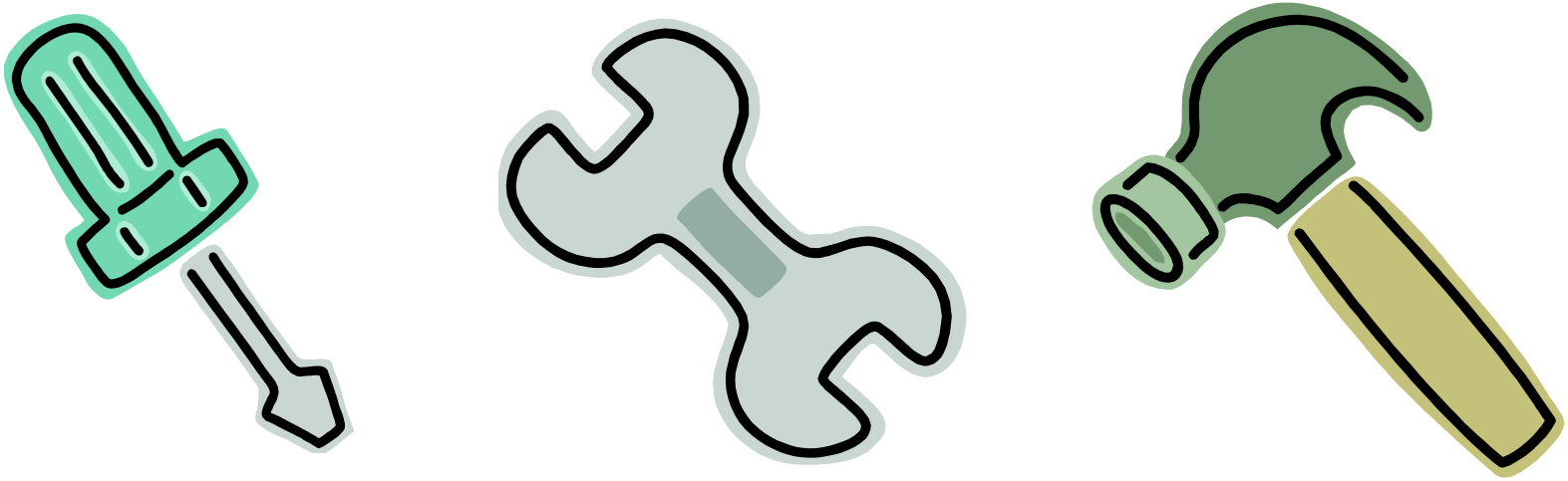
Just because the process runs,
does not mean it is a good system!

The Big 4

- Maintainable
- Reusable
- Flexible
- Automated

Maintainable

- How easily can future requirement changes be incorporated?
- Can someone with a limited understanding of the process modify the code?
- How much time is necessary to become familiar with the process before incorporating changes?



Reusable

- Share common code
- Easy access to stored routines
- Use lookups, formatting, functions

Flexible

- Plan for expansion
  - Can the process incorporate future data sources?
  - Interface with other relevant systems?

- Platform independence

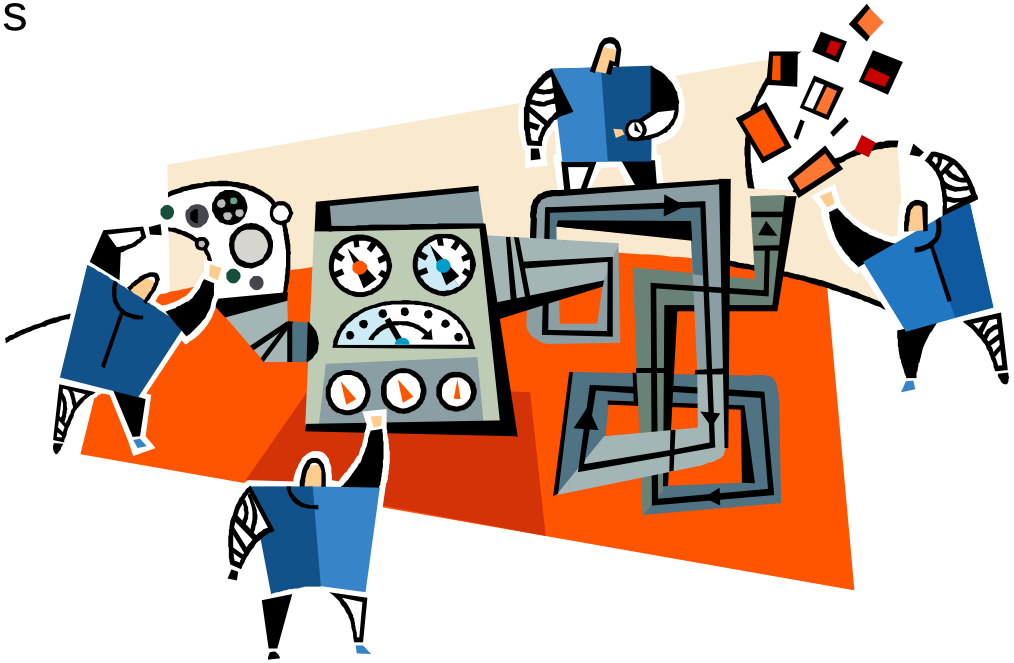
  Windows, UNIX, z/OS

- Scalability

Automated

"We have computers, we should use them."

- Minimal manual input
- Eliminate manual steps
- Make default assumptions
- Allow for override



Flexibility and Automation Example

Scenario:

- At the end of each month, a financial report is generated for the current month
- It is occasionally necessary to generate the report for a previous month
- Data reported is only for the selected month
- Some data sources contain the date as part of the source name
- The month end date is displayed in the report titles

What is a possible solution to generate the required reports?

Flexibility and Automation Example

Solution 1:

- All of the dates are hard coded into the report (titles, footnotes, data sources, calculations, etc.)
- Each month all of the dates need to be updated.
- It is easy to miss some updates and takes an hour every week to complete.

Solution 2:

- The month is a parameter that is set at the beginning of code.
- There is only one update necessary to run the report.
- The relevant date appears in the report and is used in the calculations.
- The report is accurate and quick to update.

Solution 3:

- Code the assumption that the report will be run for the system month.
- Include option to override with a manual parameter.

What solution is the BEST choice?

Solution 3

Automated and Flexible!

Flexibility and Automation Example

```
/*************************************************************/
/* if the time frame to be processed is the most recent month, then */
/* the macro variable m_timeframe is a value of current and the     */
/* process dates are set based on the system date.                  */
/*************************************************************/
%if &m_timeframe = current %then %do;
  process_mth_beg = intnx('month',"&sysdate"d,-1);
  process_mth_end = intnx('month',"&sysdate"d,-1,'e');
%end;
/*************************************************************/
/* if the time frame to be processed is not the most recent month,  */
/* then the macro variable m_timeframe is a value other than        */
/* current and the process dates are set based on the macro         */
/* variables mbegindate and menddate.                               */
/*************************************************************/
%else %do;
  process_mth_beg = "&mbegindate"d;
  process_mth_end = "&menddate"d;
%end;
```

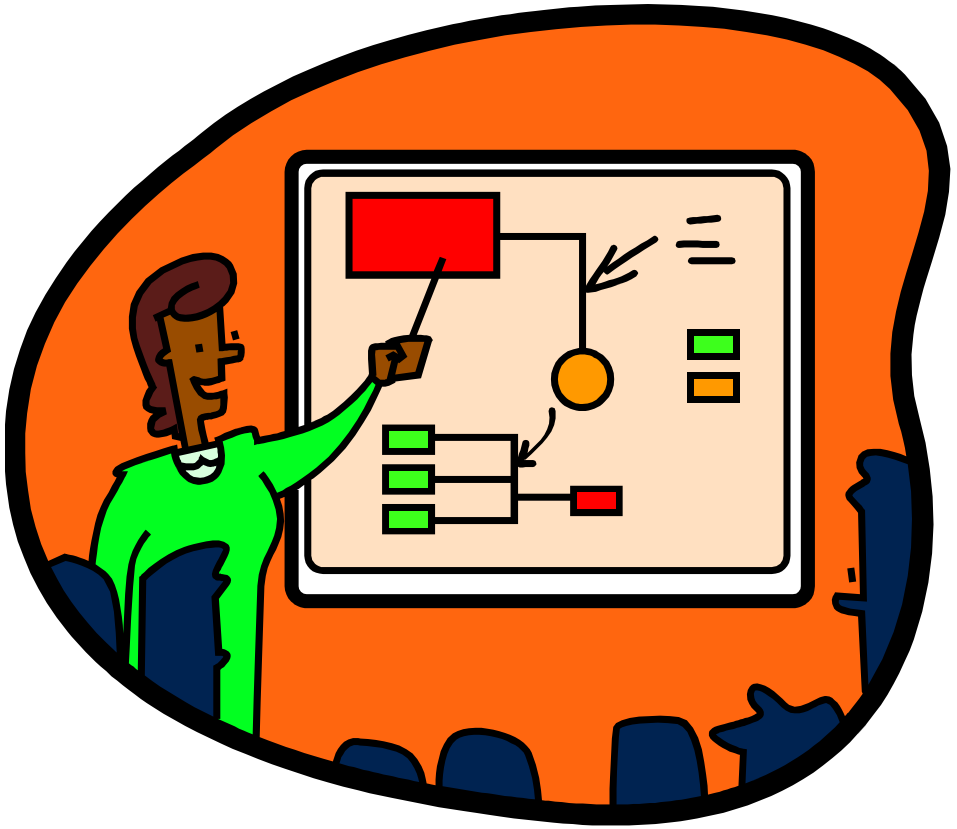
Flexibility and Automation Example

```
  /*************************************************************/
  /* set the first possible date for the history period which will   */
  /* be included on the new version of the data set when the         */
  /* processing is complete.  Any observations on the input version  */
  /* of the data set with a file_dt prior to the hist_start date     */
  /* will not be included on the new version.                        */
  /*************************************************************/
  hist_start = intnx('month',process_mth_beg,-12);
 /*************************************************************/
  /* set the last date prior to the new period to be added to history*/
  /* this date will be used to insure double of the period to be     */
  /* appended to history will not be added.                          */
  /*************************************************************/
  hist_end = intnx('month',process_mth_beg,-1,'e');

  numdays = intck('days',process_mth_beg,process_mth_end) + 1;

  call symput ('m_beg',put(process_mth_beg,date9.));
  call symput ('m_end',put(process_mth_end,date9.));
  call symput ('m_histstart_dt',put(hist_start,date9.));
  call symput ('m_histend_dt',put(hist_end,date9.));
  call symput ('m_numdays',put(numdays,3.));
Run;
```

What Makes a Good System?

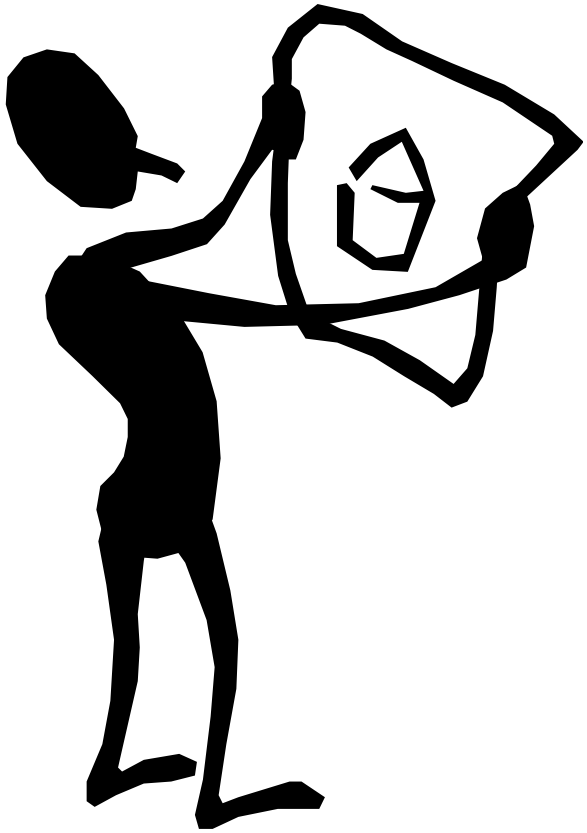
But, there is more…

- Planned
- Documented
- Restart Capable
- Efficient
- Simple
- Modular
- Accurate
- Validated
- Change Control



Planned

“Good specifications make good systems.”



- Clear and concise
- Purpose of project
- Define the scope
- List assumptions
- Deliverables-time, ownership
- Cost
- Schedule
- Change control
- Ask questions!

Planning: A Summarizing Problem

Every time First Department Store makes a sale, they record Store, Department, Amount, and Coupon in a SAS dataset INVNTORY.

First Department Store

OBS	Store	Dept	Amount	Coupon
1	101	SHOES	14.22	.
2	101	SHOES	22.85	.
3	101	SPORTS	17.11	0.99
4	102	SHOES	26.78	4.99
5	101	SPORTS	11.97	.
6	102	SHOES	54.22	12.99
7	102	SPORTS	11.16	.
8	103	SHOES	41.22	.
9	103	SPORTS	13.78	2.89
10	102	SHOES	13.72	4.49
11	101	SHOES	21.75	.
12	101	SPORTS	12.71	.
13	102	SHOES	21.78	1.88
14	103	SPORTS	51.57	6.99
15	101	SHOES	54.25	.
16	102	SPORTS	11.56	0.49
17	101	SHOES	41.52	.
18	101	SPORTS	15.79	.

Planning: A Summarizing Problem

As an analyst for First Department Stores, you are asked to create a set of analysis reports for management. Using the data from the previous slide, how could you approach the request?

What is the purpose of analyzing this data?

What assumptions can we make?

What questions to we have to ask?

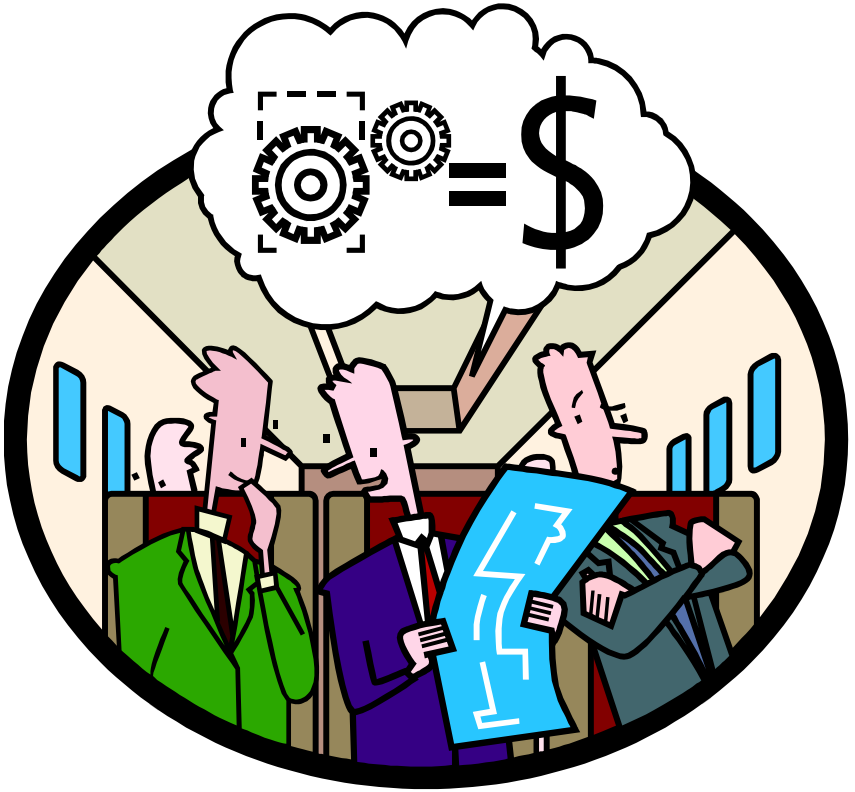
1. What numbers do we want to analyze?

2. How do we classify the data?

2. Which statistics do we want?

Planning: A Summarizing Problem

- How will we accomplish our goal?
- How long will it take?
- Who will do it and when?
- How much will it cost?
- How will we integrate changes?



Documented

Why should you document projects?

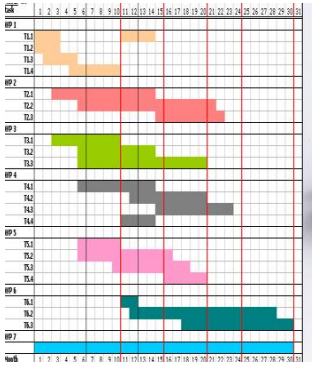
- Easier to understand
- Easier to debug and fix errors
- Easier to maintain
- Easier to make updates and changes
- Easier to pass it along or share with someone else

Documentation

- Essential component of any good system, but often overlooked.
- It is important to document what is being done, when, by who, and why
- Documentation should be included for both coding and point and click tools.
- What's obvious to you now isn't later or to someone else.

Documentation - Components

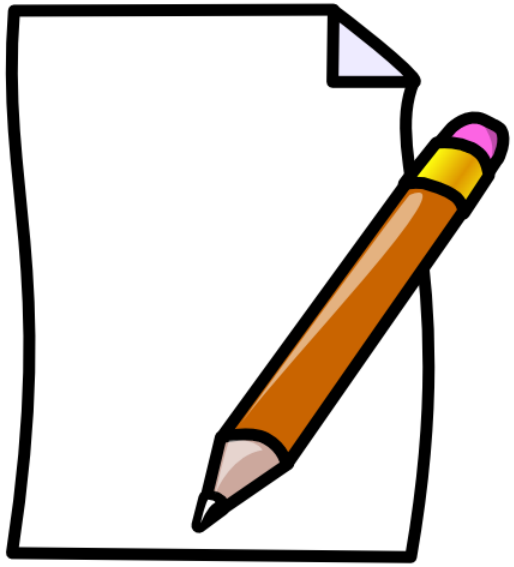
- Written specification document
- Meeting notes and status reports
- Code documentation
- Change logs
- System documentation
- Run time instructions
- Data flow



Documentation

- Document overall pieces of your Project.

- Explain the purpose of multiple tasks working together, including dependencies, and the overall purpose of the process.



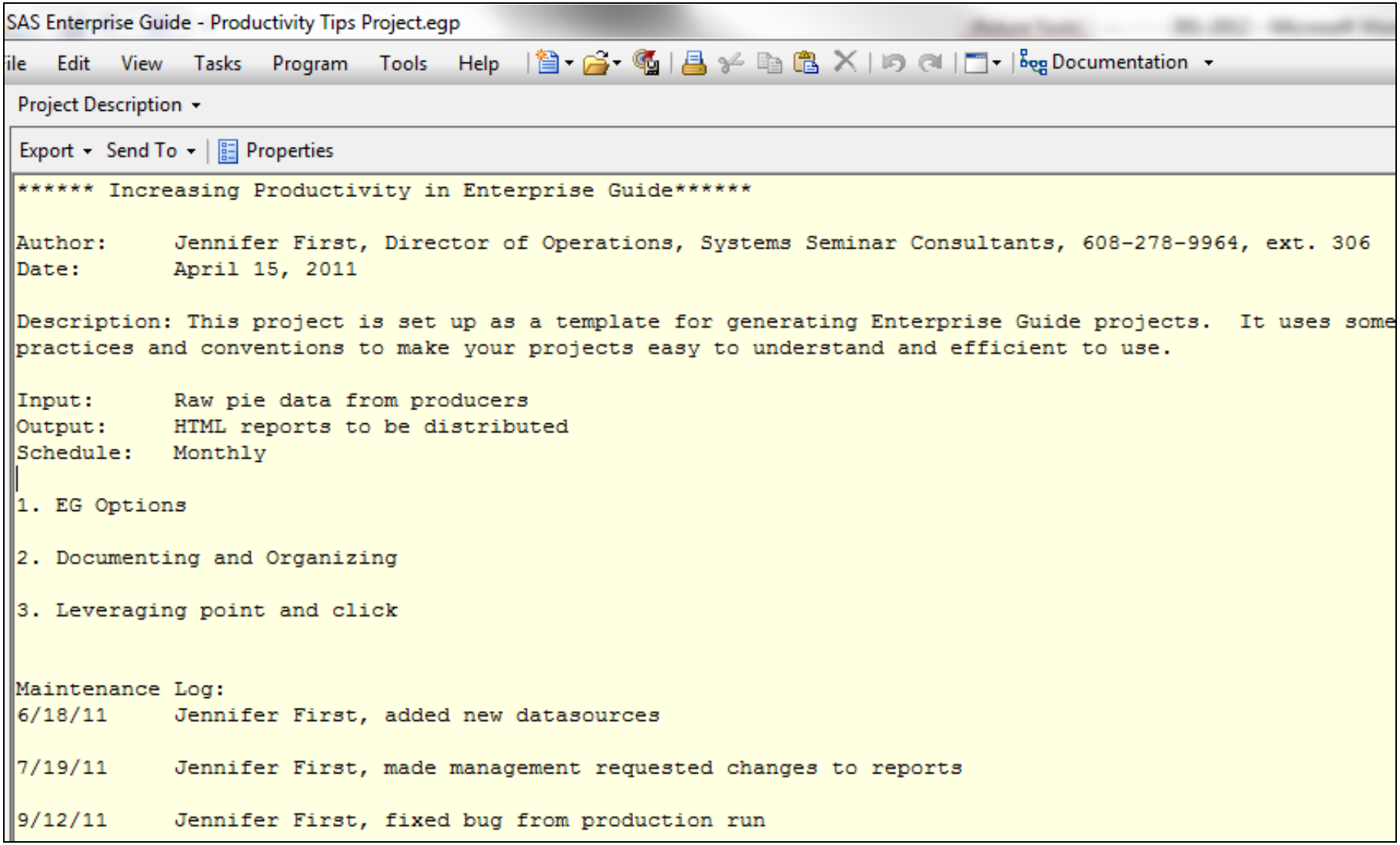
Naming Conventions

- Good, consistent naming conventions are self documenting.
- Choose meaningful names, assign variable labels, and use good default formats for numeric fields.
- This applies to data, variables, output, tasks, and programs.

Code Documentation

- Include documentation in your code!
- Use a header box and change log for each program.
- Comment as you develop code (or even before) but don't wait until after.
- Especially focus comments on difficult code, such as complicated calculations.
- Remember someone unfamiliar with the technology may need to read and decipher this code.

Code Documentation



Documentation and Organization

- Ad-hoc processes - may need a similar process down the road
- Easy to take shortcuts in the short term to rush and get something done.
- Small amount of time upfront will yield great results long term.

Restart Capable

HELP! The system crashed during my job…

Questions to consider during design:

- Does the complete job need to be executed from the beginning?
- Is there data to be rolled back to an earlier version?

How to:

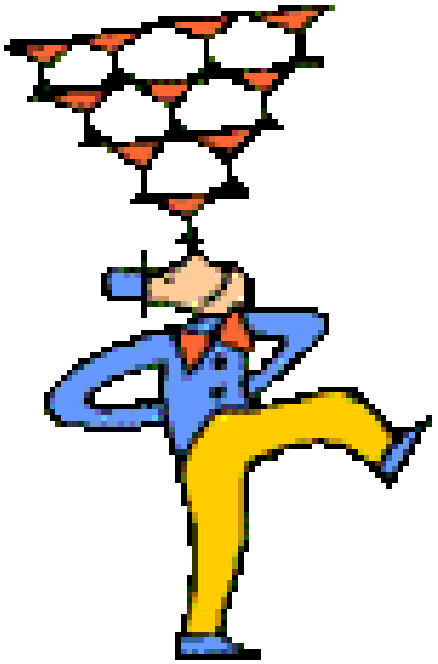
- Define distinct points for restart
- Separate job into distinct processes
- Define clear instructions for restart
- Use relative data references (GDG) to allow roll-back of data

Efficient

- Runs in a reasonable amount of time – not just fast
- Often one or a few steps take majority of the time
- Look at logs to determine where there are efficiency issues

Need to find a balance

CPU vs. I/O vs. Clock time



Keys to Efficiency

Efficient programs:

- Keep data sizes to a minimum*
- Keep data passes to a minimum*
- Use efficient data types
- Read and write selectively
- Make efficient use of coding statements
- Know defaults and override them when desirable
- Avoid sorting
- Take advantage of data characteristics

Notes:

* These two items matter more than anything else!

Avoid Sorting

Sorting can often be avoided with planning.

Tips:

- If several steps require data sorted in the same order, group those together rather than re-sorting the data each time
- Only sort data when necessary
- Remove unnecessary variables and observations
- Take advantage of sorted input files
- Use indexing when appropriate

Indexing Advantages and Disadvantages

Indexing stores pointers to locate observations more quickly and efficiently.

Advantages:

- Indexing allows quick access to a small subset of observations.
- Values returned from the index are returned in sorted order.
- SAS automatically decides whether or not to use an index.

Disadvantages:

- CPU and I/O are intensive in order to create and maintain the index.
- Extra memory is required to load the index table.
- Extra disk space on a separate file is required to store the index's data structure.
- Extra resources are required to rebuild indexes for a new version of a data file with indexes.

Indexing Guidelines

Some rules of thumb:

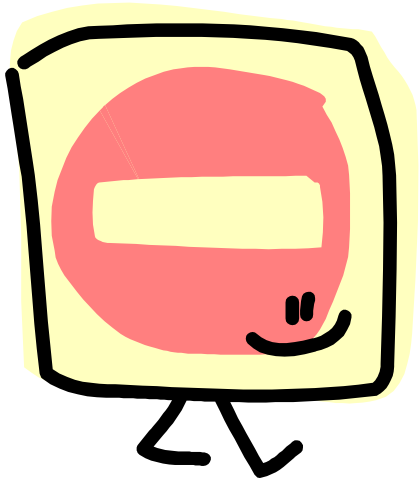
- Minimize the number of indexes to reduce the disk storage and update costs.
- Create indexes only if the data file's physical size is large.
- Indexing works best when retrieving a small number of observations relative to the total number of observations (<15%).
- Indexing works best on values not frequently updated and values that are uniformly distributed.
- Works well when the data set is subsetted by values of the index variable.

Space Issues

We frequently run out of disk space, and when processing large files, our SAS jobs fail.

Possible causes:

- The data is too big.
- Unnecessary temporary files are being kept.
- Unnecessary records and variables are present.
- Multiple copies of a file are kept while recreating or sorting.
- Variable widths are longer than necessary.



Space Issues

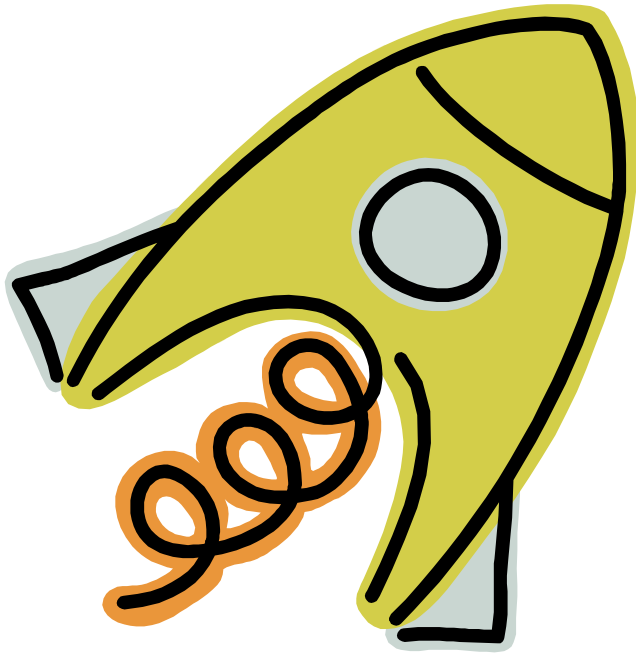
What can we do about it?

Possible solutions:

- Make the libraries larger.
- Use an alternate temporary or permanent disk library.
- Use tape datasets instead of disks.
- Clean up the program to avoid carrying any unneeded data.
- Understand and minimize sorting space demands.
- Use logical *views* of the data instead of physical data files.

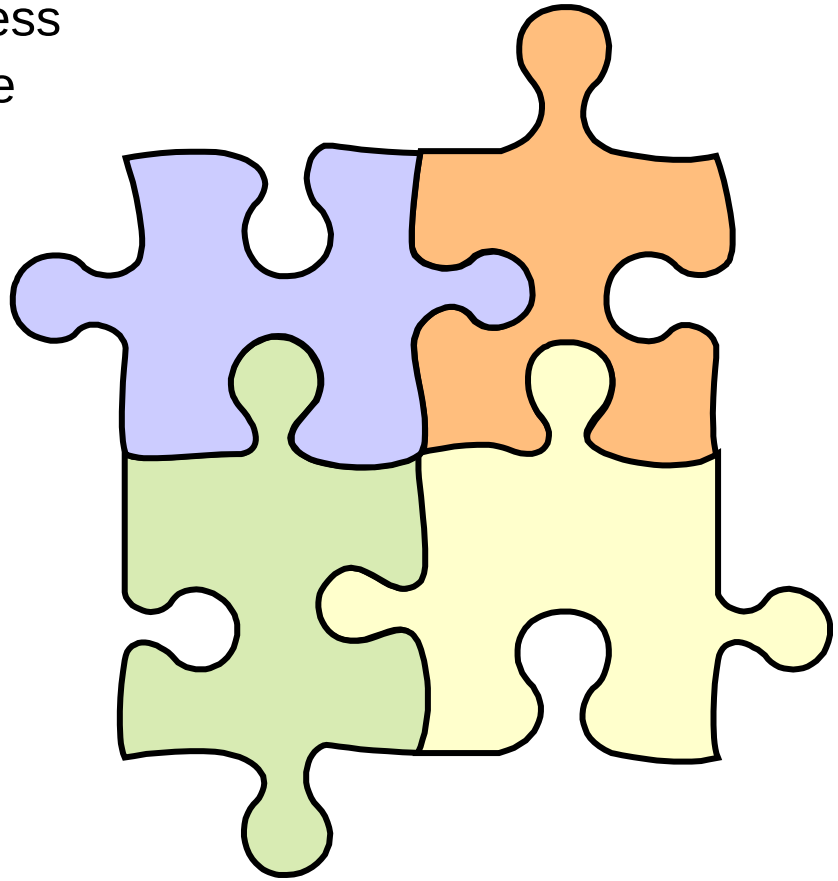
Simplicity

- Reasonable person could understand and work on it.
- “It’s not rocket science.” (although sometimes it is…)
- Minimize complicated processes like macros.



Modular

- Allows measureable goals and objectives
- Easier to proactively address issues
- Fewer surprises at the end of a project
- Simplifies restart process
- Simplifies maintenance



Modular

Logical Breaks

- Break up a Financial Projection system, into 3 modules:
  - Income Projections
  - Expense Projections
  - Production Projections

- Each section has its own data, code, and output

- Easy to keep track of and make changes to

- Works well for a process that has logical pieces to it

Modular

Functional Breaks

System level and how the process will be run.

1. Overall System Documentation
2. Startup Processes, including Assigning Libraries and Setting Options
3. Data Import, Data Cleanup, Data Preparation
4. Preliminary and Exploratory Analysis
5. Final Analysis
6. Graphing and Reporting
7. Production, Scheduling, and Distribution

Accurate

- Uses the right data
- Accurate calculations
- Audits in place to ensure accuracy

Validation

- Manual compares
- Automated compares
- Independent parallel system

Change Control

How are future requirement changes implemented?

- Document request
- Code changes occur in a test environment
- Perform testing and validation in the test environment
- Coordinate migration to the production environment
- Update system documentation

Conclusion

- It works
- Easy to use and maintainable
- Documented
- Customer (external or internal) needs come first
- Focused on the long term

A good measure of a good system is no call backs

Questions, Comments

- Please email questions or comments to train@sys-seminar.com or contact us at 1-800-997-7081.

- A copy of the presentation and a recording of the webinar will be emailed out to attendees. This can be shared with people who did not attend.

Upcoming Lunch and Learn

SAS Efficiencies

September 13, 2012

Thomas Miron

For registration, email train@sys-seminar.com

Contact Us

SYSTEMS SEMINAR CONSULTANTS, INC.

SAS® Training, Consulting, & Help Desk Services

2997 Yarmouth Greenway Drive • Madison, WI 53711

(608) 278-9964

www.sys-seminar.com

sas SAS Alliance *Silver Member*

Teresa Schudrowitz
Senior Consultant
tschudrowitz@sys-seminar.com

Jennifer First
Director of Operations
jfirst@sys-seminar.com