Presentation
Paper
Bio

P R E S E N T A T I O N

T13

Thursday, February 18, 1999
1:30PM

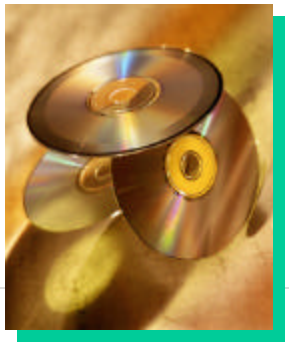
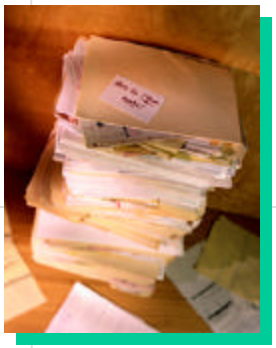
TOOLS OF THE TRADE: HOW TO AUTOMATE A SOFTWARE REPOSITORY

James Heires
Rockwell Collins, Inc.

International Conference On
Software Management &
Applications of Software Measurement
February 15-19, 1999 • San Jose, CA

Tools Of The Trade

How To Automate
A Software Repository

James T. Heires
February 18, 1998

Rockwell Collins

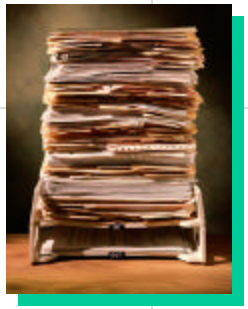
Agenda

- Background
- Process Automation
- Convincing Your Audience

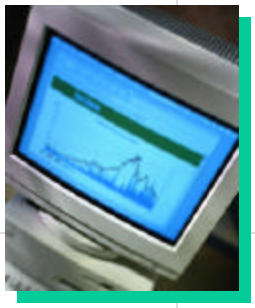
Background

- Benchmark

- 200 Added Projects



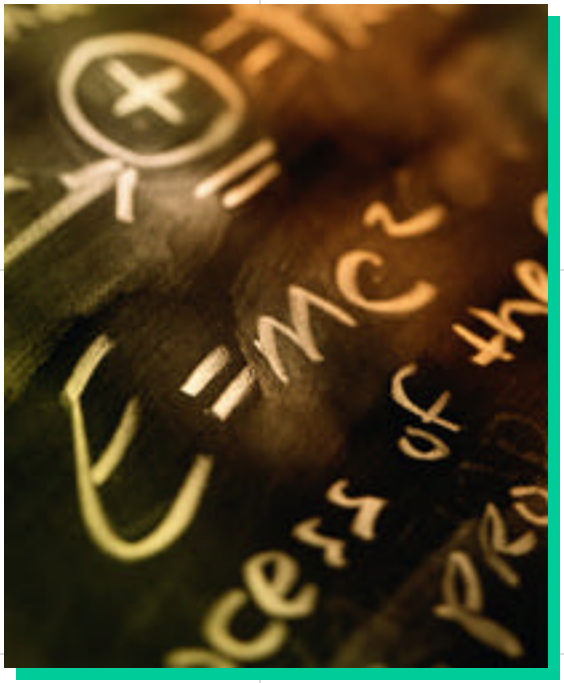
- Use Of Repository



“Those Who Cannot Remember The Past Are Condemned To Repeat It”

George Santayana

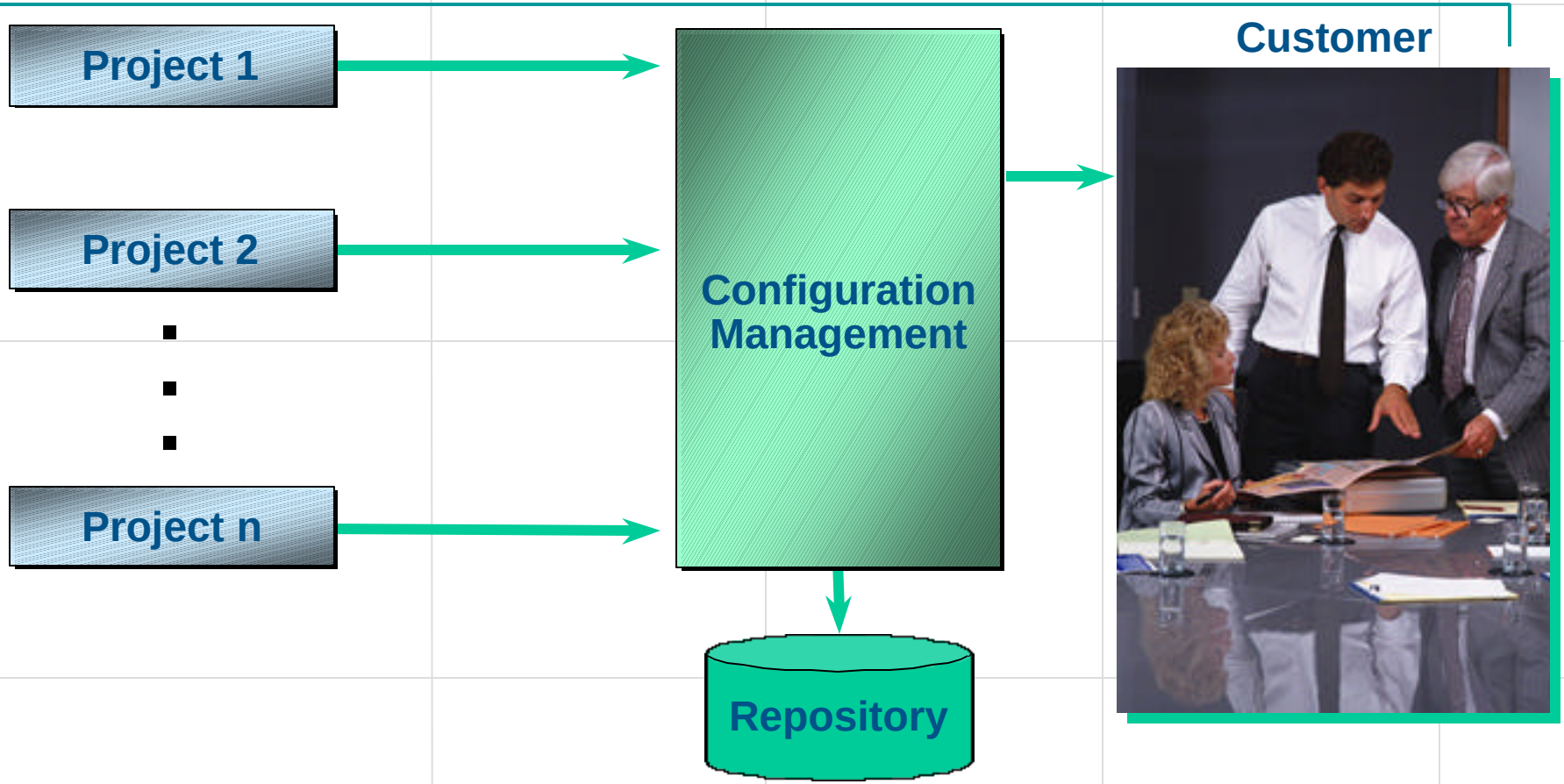
Simple, But Not Easy



How We Did It

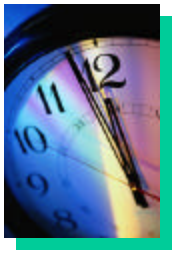
Data Collection Process

Configuration Management

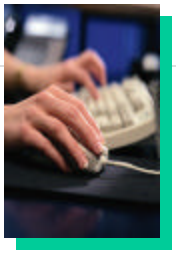


Rockwell Collins

Data Collection



Time



Effort

Attributes

User Interface

Repository

Validation

Independent Reviewer

- Consistent
- Complete
- Correct
- Confident

Analysis

Compare Project To:

- Product Family
- Business Unit
- Competition

Analysis

MS Access Query

HDC Summary : Select Query

Schedul	Effort	Dev Defec	ESLOC	BU Avg Sch	BU Avg Effo	BU Avg Dev Defe	BU Avg ESLO
0.07	8		76	4.89	246.00	60.9	11001
0.16	8		330	4.89	246.00	60.9	11001
0.10	8		178	4.89	246.00	60.9	11001
0.10	8		97	4.89	246.00	60.9	11001

Record: 3 of 395

Mail-Merge

	Project	Product Family	Business Unit	Industry
Size				
Effort				
Schedule				
Defects				

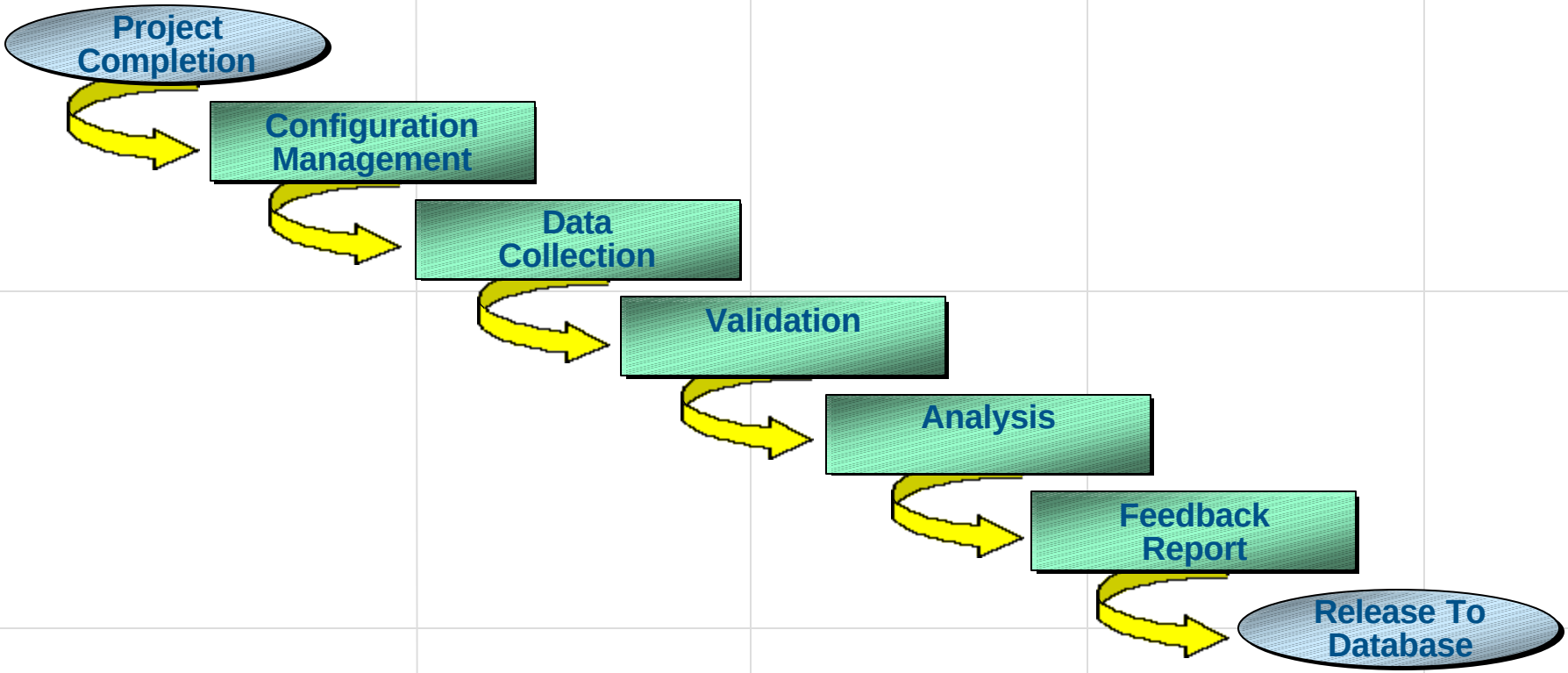
MS Word Document

Rockwell Collins

Feedback Report

- Analysis
  - Summary
  - Details
- Recommendations

Data Collection Process

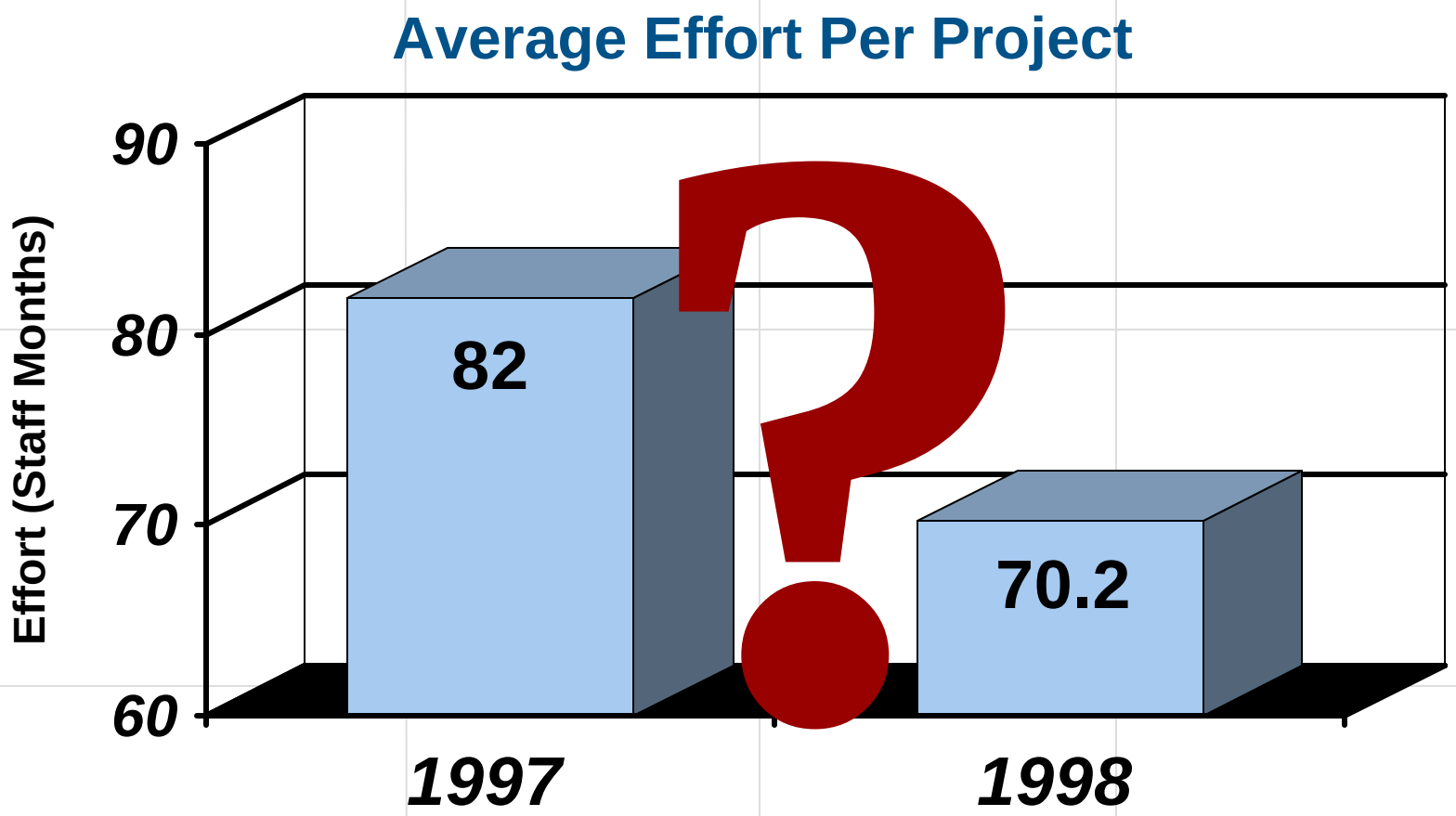


Rockwell Collins

Convincing Your Audience

- Student's T-Test
- Control Charts
- Capability Studies

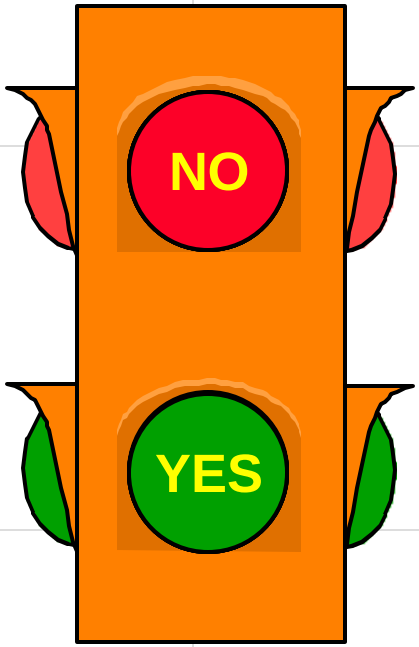
Convincing Your Audience



Rockwell Collins

Convincing Your Audience

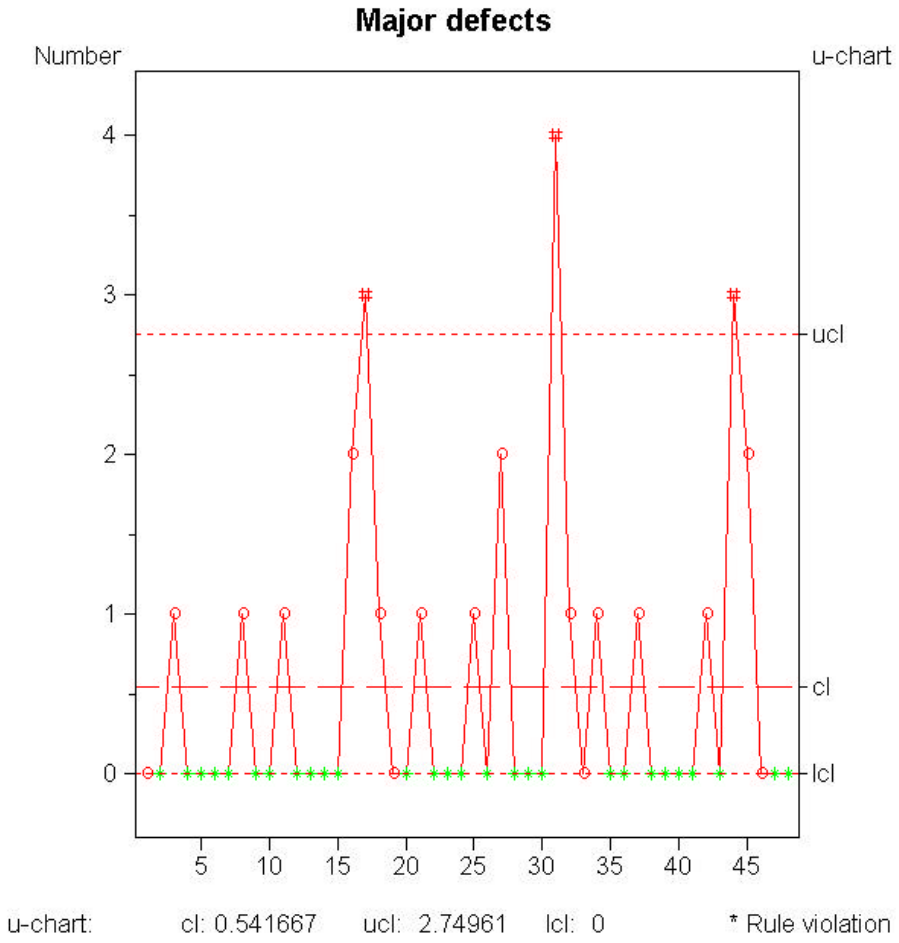
Student's T-Test



Rockwell Collins

Convincing Your Audience

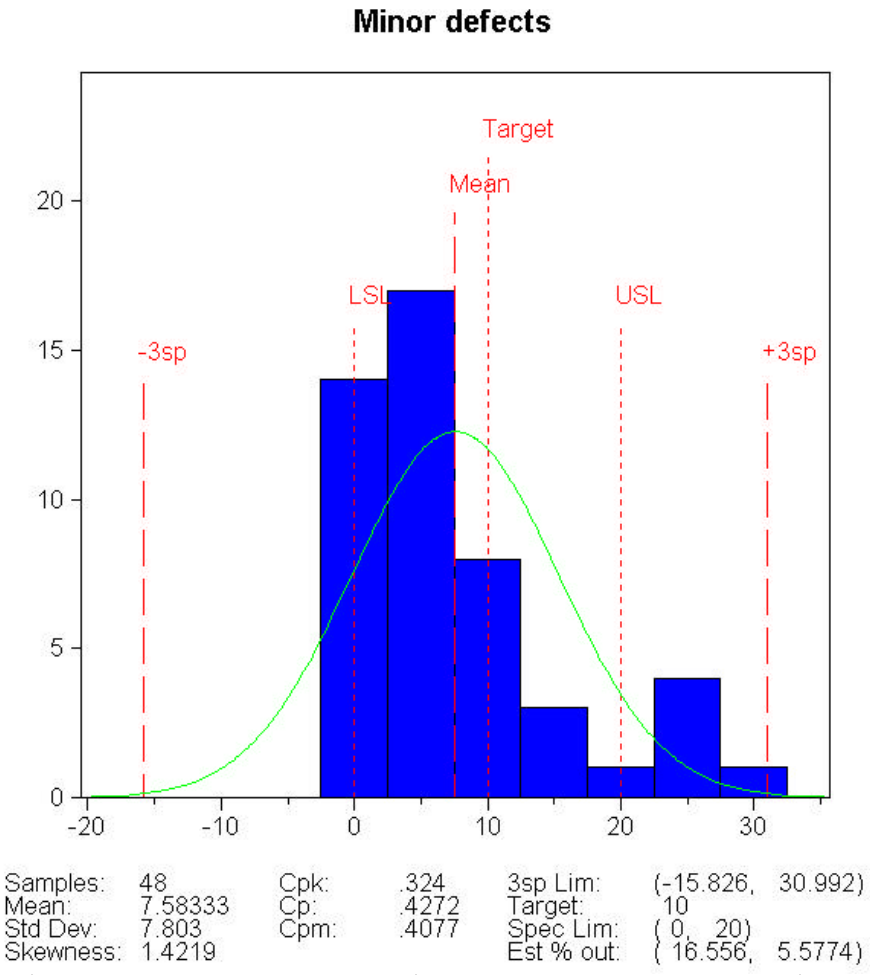
Control Charts



Rockwell Collins

Convincing Your Audience

Capability Studies



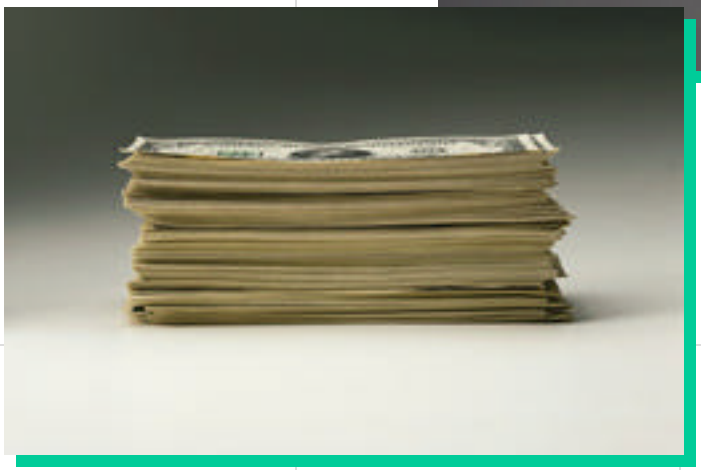
Rockwell Collins

A Few Words About Program Success

- Support Business Goals
- Start Simple
  - Minimum Data Set
  - Few Projects
- Quick Success Story

Provide Value Or Perish

- Customer Focus
- Return On Investment



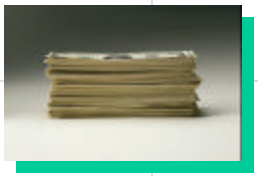
In Conclusion . . . .

- Support Business Goals

- Automate

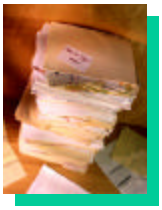
- Measure Program Performance

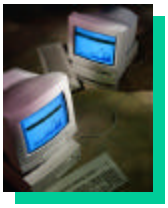
- Provide Value Or Perish



Tools Of The Trade

How To Automate
A Software Repository



James T. Heires

Abstract:
Building a historical software repository is critical to the success of cost estimation, process improvement and metrics programs. Without a repository, there is no effective way to sanity-check estimates or measure improvements. Actually collecting data, however, is not easy. Organizational and technical issues impede efficiency and consistency needed for a first-rate repository. Automation can help solve these issues by greatly increasing the quality and pace of the data collection activity. This paper describes a case study of the Rockwell Collins approach to repository automation, specific examples of tools implemented and benefits realized through automation.

Those who cannot remember the past are condemned to repeat it.[1]

Background:
Rockwell Collins began building its corporate software repository in 1994. A 16-project benchmark study carried out by QSM Associates, Inc. established a corporate capability baseline to measure improvements. In addition, the benchmark was compared with industry averages to establish the competitive position of Rockwell Collins. Since 1994, historical data from over 200 projects has been added to the repository. Two full-time and many part-time staff members were required to build and execute the program. Three University of Iowa summer interns implemented the automation described in this paper. The repository was used to evaluate the impact of process improvement efforts and to support achievement of SEI CMM[2] Level III. Estimates of new software development projects were also based upon repository data. The benefits of a successful repository are apparent, but they do not come for free.

Simple, but not easy:
A starting set of core metrics constitutes only a handful of simple measures[3], but building and maintaining a historical repository can be very difficult. Organizational challenges such as funding, staffing and achievement of business goals are vital to success but are beyond the scope of this paper (see Romanowsky[4] for a discussion of these challenges). The technical challenges, however, of selecting and implementing the most beneficial automation are addressed. The vendor-supplied benchmark study jump-started the repository by specifying a database and a methodology. The highly flexible QSM database is built upon Microsoft® Access, which makes the collection process easier to automate[5].

The Process:
The first step of the collection process occurs after the completion of development work. Rockwell Collins has a configuration management (CM) office that archives and catalogs all software after development completes, which provides a convenient point to collect data. Source lines of code (SLOC) is the primary measure to collect at this point.

The traditional approaches to size counting within Rockwell Collins represented a large source of variation. The numerous definitions of SLOC and the proliferation of homegrown counting tools attributed to the variation. An automated line counting utility was integrated into the existing tool suite operated by the CM office. The result of this size measurement is the count of new, modified and unmodified SLOC. Important aspects of a size counting tool include:

- The use of industry standard definitions[6]
- The support of multiple languages and platforms
- The auto-detection of source language
- The measurement of lines modified from a previous version

Next, project performance data is collected from the project leadership. Time and effort are the primary quantitative values to collect, but qualitative attributes (e.g., application type, customer type and programming language) are also sought. The electronic data collection program provided by QSM simplifies this step. This program is the data entry interface of the repository. By using this program, consistency in data definitions and attribute classifications was directly achieved. It also eliminated the recurrence of manual data entry, indicative of using paper forms. The data entry application was distributed over the nationwide computer network at Rockwell Collins to simplify remote data collection as well.

After collection, the data is validated to ensure consistency, correctness and completeness. Only independently validated project data is allowed into the repository. This minimizes dysfunctional behavior by project personnel attempting to "game" the system to look good. Validation consists of a 30- to 60- minute interview with project leadership. Sources of data, incomplete information, lessons-learned and significant factors are discussed in the interview. Michael Mah calls these interviews "mini post-mortems" in his article on software metrics.[7] Once a level of confidence about the validity of the data has been reached, analysis can begin.

Analysis of project data compares the target project to similar projects in the repository. Size, time, effort and quality are compared quantitatively across the following domains:

- Product family
- Business unit
- Industry averages

The comparison is summarized on the first page of a feedback report that goes to project leadership. A detailed analysis appears later in the report. This comparison was time-consuming, which made it a prime candidate for automation.

Tools to accomplish this analysis include database queries coupled with the Microsoft Word mail-merge feature. Figure 1 illustrates this concept. The Microsoft Access query shown is populated with project and business unit data from the repository (project on the left and business unit on the right). The project portion of the query simply regroups appropriate data from individual project records. The business unit portion of the query summarizes data from projects belonging to the same business unit. Product family and Industry data is similarly populated, but not shown here. The Microsoft Word document represents a portion of the feedback report and imports data from the query in a meaningful fashion.

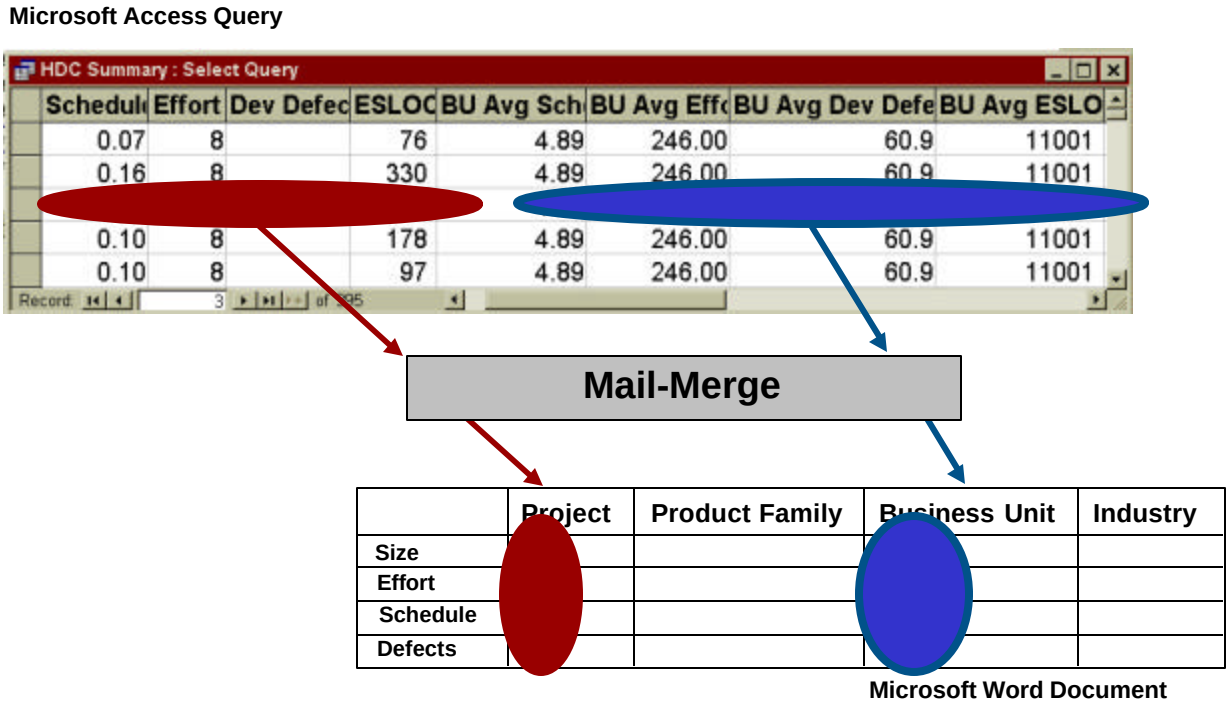


Figure 1 – Feedback Report Automation

To import the data from the query to the report, open a document created from a prepared template that is connected to the query, then select the name of the project to analyze. All the data from the query is automatically mail-merged into the document like a form letter.

This feedback report serves to confirm the importance of the data collection program and to provide a simple analysis of project performance in comparison with similar projects.

After the feedback report has been completed, the data is released to the repository. The data is then available to support estimation and other analysis activities—ending the collection process.

The data collection process is summarized in Figure 2 below:

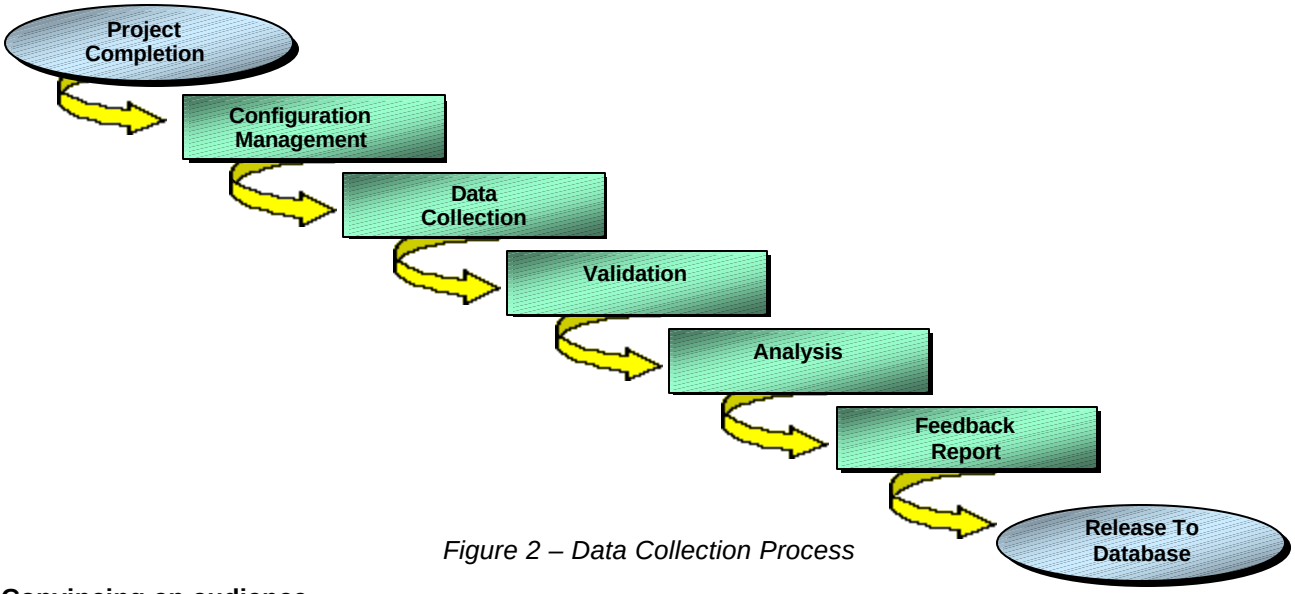


Figure 2 – Data Collection Process

Convincing an audience:
Simple analysis of a project against average performance of a domain is satisfactory for feedback and validation, but not when attempting to influence pivotal business decisions! Fortunately there are several sophisticated statistical tools available to help.

The student's T-test, control charts and capability studies can be used to determine:

- If significant change has occurred
- if a process is in statistical control
- if business goals are being achieved

The Student's T-test:
The student's T-test is based upon a t-distribution. The test can be used to make inferences about a population mean[8]. First, start with a null hypothesis (usually the mean of one sample is equal to another sample mean). The results of the test determine whether to accept or reject the hypothesis based upon the results of the test. As an added bonus, the student's T-test accounts for small sample sizes.

Suppose a senior manager wants to know if the development cost (measured in staff-months) of software has decreased since last year. The first method that comes to mind might be to measure and compare the average cost for all projects in each of the two time periods (last year and this year). If the average cost of this year is less than last year, one might claim an improvement. The magnitude of the improvement is the difference between the two average values (see Figure 3 below). However, if the averages are similar, sample sizes are small or variation is large, confidence in the analysis falters.

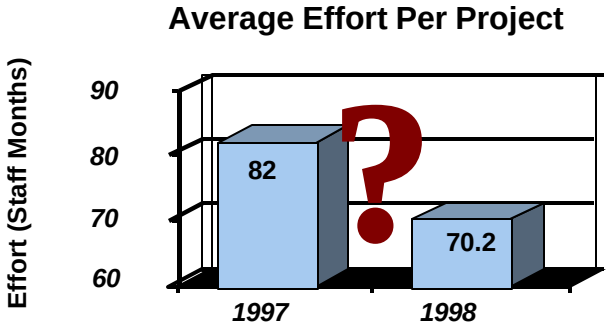


Figure 3 – Simple Analysis

At this point, student's T-test comes in. This same set of data analyzed with the T-test shows no significant cost difference between the two years.

The T-test also provides a confidence level result to further understand the underlying data. If an important business decision was dependent upon the results of this analysis, costly mistakes could be avoided by using this test.

Control charts and capability studies:
Control charts are used to determine if a process is in statistical control (e.g., predictable), and who should be assigned to address process issues. The applications of control charts to software engineering are many and varied. Some examples include:

- determining the impact of process improvement efforts on productivity[9]
- understanding whether the defect detection process is consistently finding defects (see Figure 4)
- whether management or engineering should be called upon to address a perceived issue with project cost.

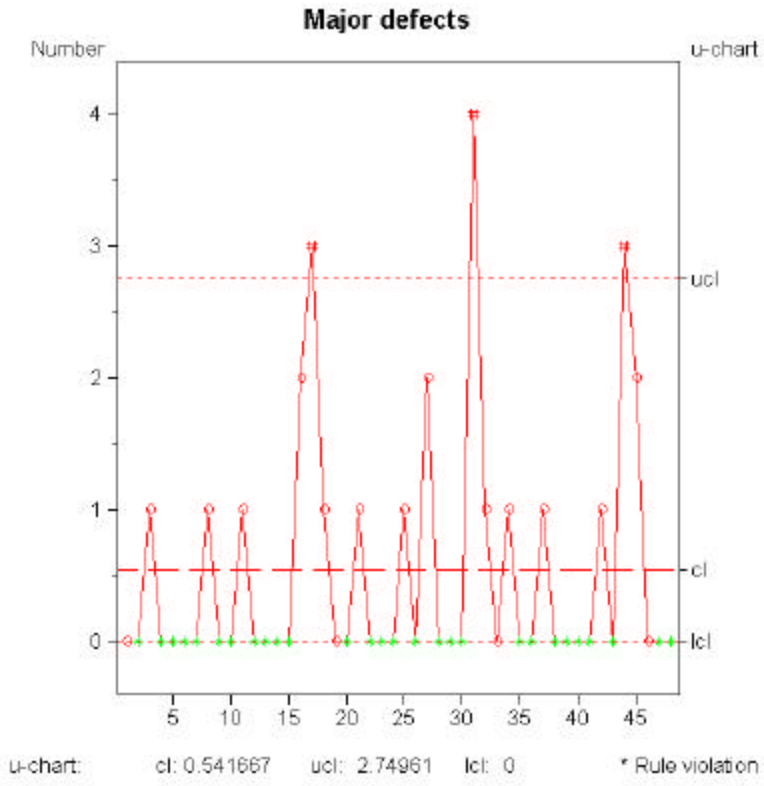


Figure 4 – *Control Chart*

Capability studies are used to determine whether a process is able to produce product within specifications (e.g., capable) and what percent of product will meet specifications in the future. Some examples of capability studies in software engineering include:

- projecting the reliability of an incomplete product based on defects detected to date (see Figure 5)
- determining whether an underway project will meet its cost and/or schedule budgets
- understanding what portion of a current product meets customer expectations

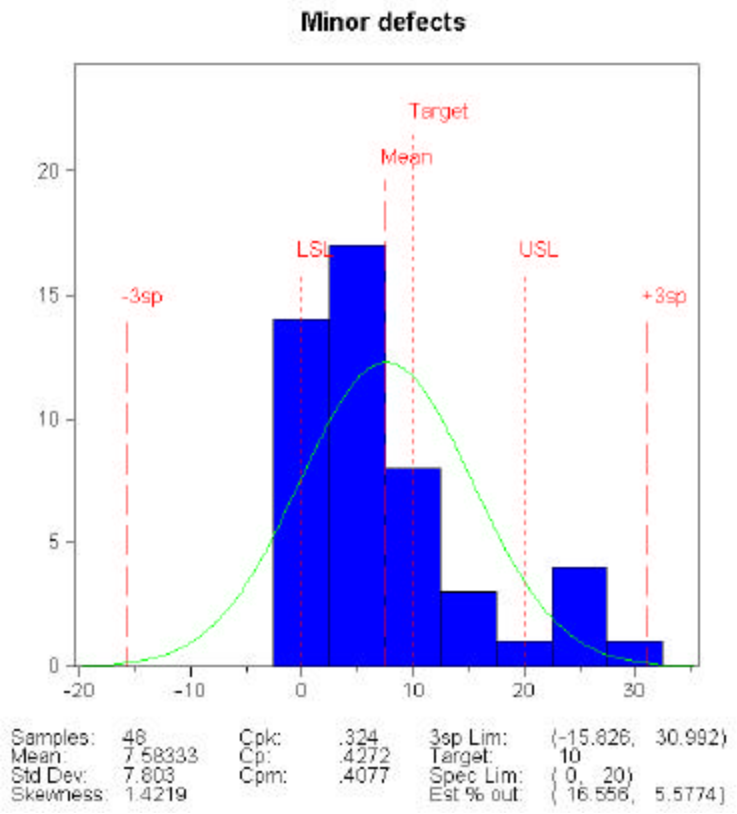


Figure 5 – *Capability Study*

This short illustration of the use of statistics in software engineering is meant to highlight a few ways to analyze data from a repository. These techniques (if correctly used) can increase confidence and convince an audience. See the references to learn more about statistical process control [10, 11, 12].

A few words about program success:

To ensure success, a repository program should consider a simple start*:*

- Concentrate on a minimum data set and a small, but significant set of projects
- Make a strong connection to business goals
- Achieve a quick success story to catapult your program to higher levels of success

Provide value or perish:

Data collection programs are doomed to failure if they do not provide value to customers. The program personnel must continually measure the value they provide from the customer's point of view, quantify this value and produce a return on investment ratio (ROI) to share with stakeholders.

The investment portion of ROI includes all costs to establish and maintain the program. This information is normally available from accounting or time-keeping records.

The benefit portion of ROI is more difficult to determine, but should only include direct benefits. This avoids the embarrassing scenario of having to wave one's hands excessively while explaining. Direct benefits typically fall into the cost-avoidance category. Some direct benefits include:

- avoiding underbidding a new program (and leaving money on the negotiating table)
- avoiding lost contracts due to overbidding or CMM requirement underachievement
- reducing the cost to produce an estimate
- avoiding the unnecessary purchase of tools

Conclusion:

A software repository program can truly thrive by connecting business goals with program goals, measuring program benefits and minimizing repository-related expenses through automation. A program should strive to select the most beneficial automation, analyze repository contents to find truth and provide value at a reasonable cost.

About the author:

James Heires is a 13-year veteran of the software industry. His current professional interest lies in engineering cost estimation using historical project data. Contact Mr. Heires via e-mail at jtheires@collins.rockwell.com

About Rockwell Collins:

Rockwell (NYSE:ROK) is a global electronic controls and communications company with leadership positions in industrial automation, avionics and communications, and electronic commerce. In late June Rockwell, announced that it planned to spin off to shareowners its Semiconductor Systems business at calendar year end. Rockwell's continuing businesses will have projected fiscal 1998 sales of approximately $7 billion and 38,000 employees. Visit Rockwell Collins at http://www.collins.rockwell.com.

---

[1] Santayana, George, *The Life of Reason*, Volume 1, 1905.

[2] Paulk, Mark, et at., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Software Engineering Institute, Carnegie Mellon University, Addison-Wesley, 1995.

[3] Carleton, A., et al., *Software Measurement for DOD Systems: Recommendations for Initial Core Measures*, (CMU/SEI-92-TR-019, ADA 258 305), Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, September 1992.

[4] Romanowski, Helen, *Building A Measurement Program Foundation: "Behind the Scenes",* Applications of Software Measurement Conference, 1996.

[5] Heires, James, *Measuring QSM's repository and analysis tool,* Application Development Trends Magazine, Volume 5, Number 6, Software Productivity Group, Natick, MA, June 1998. See also: http://207.121.151.13/Pub/jun98/pr601-2.htm.

[6] Cruickshank, R. & J. Gaffney, *Code Counting Rules and Category Definitions/Relationships*, Version 02.00.04, Software Productivity Consortium, April 1991.

[7] Mah, Michael C., and Putnam, Lawrence H., *Software by the Numbers: An Aerial View of the Software Metrics Landscape,* American Programmer, ©November 1997.

[8] Ott, Lyman, *An Introduction to Statistical Methods and Data Analysis*, Duxbury Press, 1977.

[9] Bechtold, Richard, and Sheckler, John, *Quantitative Management*, Software Productivity Consortium, SPC-96020-MC, March 1997.

[10] Wheeler, Donald, *Understanding Variation: The Key to managing Chaos*, Knoxville, TN, SPC Press, 1993.

[11] Wheeler, Donald and Chambers, David, *Understanding Statistical Process Control*, Knoxville, TN, SPC Press, 1992.

[12] Wheeler, Donald, *Advanced Topics in Statistical Process Control*, Knoxville, TN, SPC Press, 1995.

James T. Heires

James Thomas Heires is a 13-year veteran of the software industry, the majority with Rockwell Collins, Inc. His professional experiences include design of consumer electronics, Electronic Flight Instrumentation Systems (EFIS), Engine Indicator and Crew Alerting Systems (EICAS) and Flight Management Systems (FMS). Five years of software process improvement followed, illuminated by the achievement of SEI CMM Level III in two Rockwell Collins business units. Mr. Heires is currently working to improve the state-of-the-practice of project cost estimation.

Mr. Heires received his bachelor's degree in Electronics Engineering Technology from the University of Nebraska and has pursued postgraduate studies in Software Engineering and Computer Science at the University of Iowa. He received recognition in 1998 from *WHO'S WHO of Information Technology*.

Mr. Heires writes software product reviews for *Application Development Trends* and delivers technical presentations at national conferences.