OVERVIEW OF THE SOFTWARE ENGINEERING LABORATORY

F. McGarry
GSFC

INTRODUCTION

The Software Engineering Laboratory (SEL) is an organization which is functioning for the purpose of studying and evaluating software development techniques in an environment where scientific application software systems are routinely generated to support efforts at the National Aeronautics and Space Administration (NASA). This laboratory has been a joint effort between NASA/Goddard Space Flight Center (GSFC), Computer Sciences Corporation (CSC), Computer Sciences Technicolor Associates (CSTA), and the University of Maryland.

PURPOSE OF THE SEL

Over the past number of years, software costs seem to have been continually increasing in relation to the costs of computer hardware. Because of the vast amounts of resources that have been directed toward the software development problem, there has been a just concern over the overall process of software development.

There certainly are many reasons for the growing concern about the software process. Not only is a sizable portion of government and corporate budgets spent on it, but systems have been getting more complex and software has been required to perform tasks previously considered unattainable. All of this has been due to the rapidly advancing technology in related fields such as computer hardware.

Therefore, in response to these problems, the science of software engineering evolved as a way of developing software through a well-defined process. Using this approach, the software process can be better understood and an attempt can be made to study and improve the product.

Great advances have been made in adding disciplines to this very young science. Over the past 10 years, the advent of disciplined design, development, methodologies, improved management techniques, software metrics and measures, automated development tools, resource estimation models, and many other approaches that have given birth to the term software engineering.

Although numerous software development methodologies have been developed, each claiming to be more effective than the other, it has not been clearly understood (at least as applied to the NASA/GSFC environment) what effects the various methodologies have on various phases of the software development process. More specifically, it has not been understood whether structured, programming, automated tools, organizational changes, resource estimation models, or any of the other technologies would have any effect (either positive or negative) on the software development process at NASA/GSFC. It has also become very clear that it is not easy to define what is a "better" software product. For these reasons (and for several others), the Software Engineering Laboratory (SEL) at GSFC was created.

The SEL set out to accomplish the following two important and valid tasks:

1. To clearly understand the software development process at NASA/GSFC (i.e., how people are used, how money and time are spent, how other resources such as the computer itself are used, how well time lines and milestones are estimated, etc.).

2. To measure the effects of various modern programming practices (MPP's) on the NASA/GSFC software development process.

In order to accomplish these goals, software which was developed for satellite mission support was studied. The Systems Development Section at NASA/GSFC is responsible for generating all flight dynamics support software for GSFC-supported missions. This software includes attitude determination, attitude control, orbit control, and general mission analysis support systems.

The SEL was then used to closely monitor all software developed to support the charter of the Systems Development Section. This includes software developed both by GSFC employees and contractor personnel (primarily Computer Sciences Corporation (CSC)). The SEL was created in the summer of 1976, and it was anticipated that the monitoring process would first of all be done on tasks using conventional means of software development, with various MPPs applied to similar tasks in an attempt to measure the effects of these practices.

Needless to say, the efforts required to accomplish the goals were far more monumental than any member of the SEL ever imagined. Extra efforts were required on nearly every plan of the experiment. Some of the underestimated areas of work include the following:

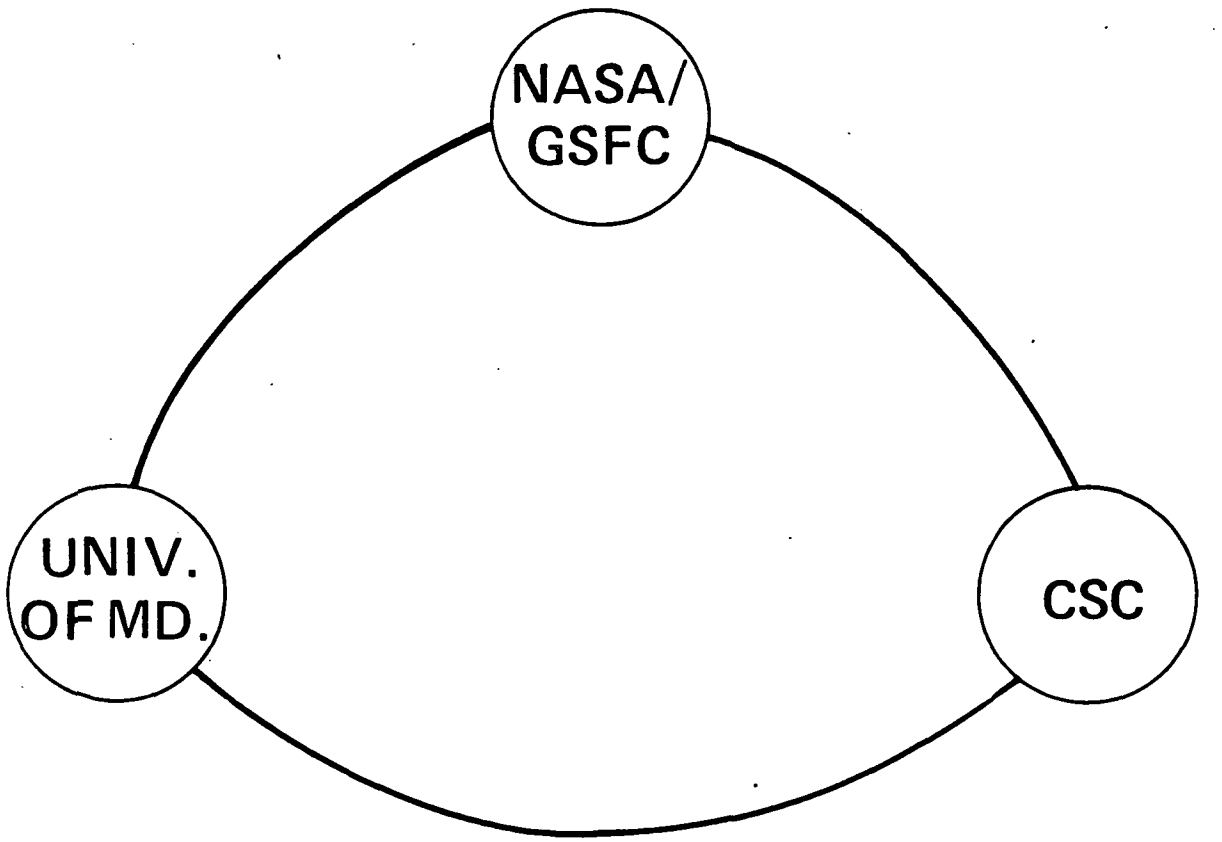
1. Development of clear, understandable data collection forms
2. Organization of the data collection process
3. Design of data storage media for data collected
4. Validation of data made available through data collection forms
5. Design of meaningful, feasible reports that could reflect early results of available data

Through all of the problems and discouraging times that the SEL experienced, the real credit for the success that the lab may have had in the past, and hopefully in the future, must go to the programmers and managers of the tasks involved in the monitoring process. Initially it was felt that a major obstacle was going to be the psychological problem of convincing programmers to accurately provide data on their efforts. However, it was found that not only did quality software people not resent providing the data but they actually made extra efforts to ensure that the data were valid and useful.

The data that have been collected by the SEL cover software development projects starting in late 1976 through 1979. It is anticipated that data will continue to be collected and studied in the future. There have been approximately 25 projects involved, ranging in size from 1500 lines of source to over 120,000 lines of source. Most of the projects were in the 40,000 to 70,000 lines of source category. All of the projects studied were development tasks used to support the flight dynamics area for the Mission Support Computing and Analysis Division (Code 580) at GSFC. The data made available to study the MPPs were collected from a series of forms which were used by all projects. In addition, data were collected through interviews, on-line accounting systems, and by personal inspections of the information by the members of the SEL.

Having investigated projects totaling somewhere around 1 million lines of code, members of the SEL feel that they have been successful in not only gaining insight into the software development process, but also in determining the relative effects of various techniques applied to the software projects.

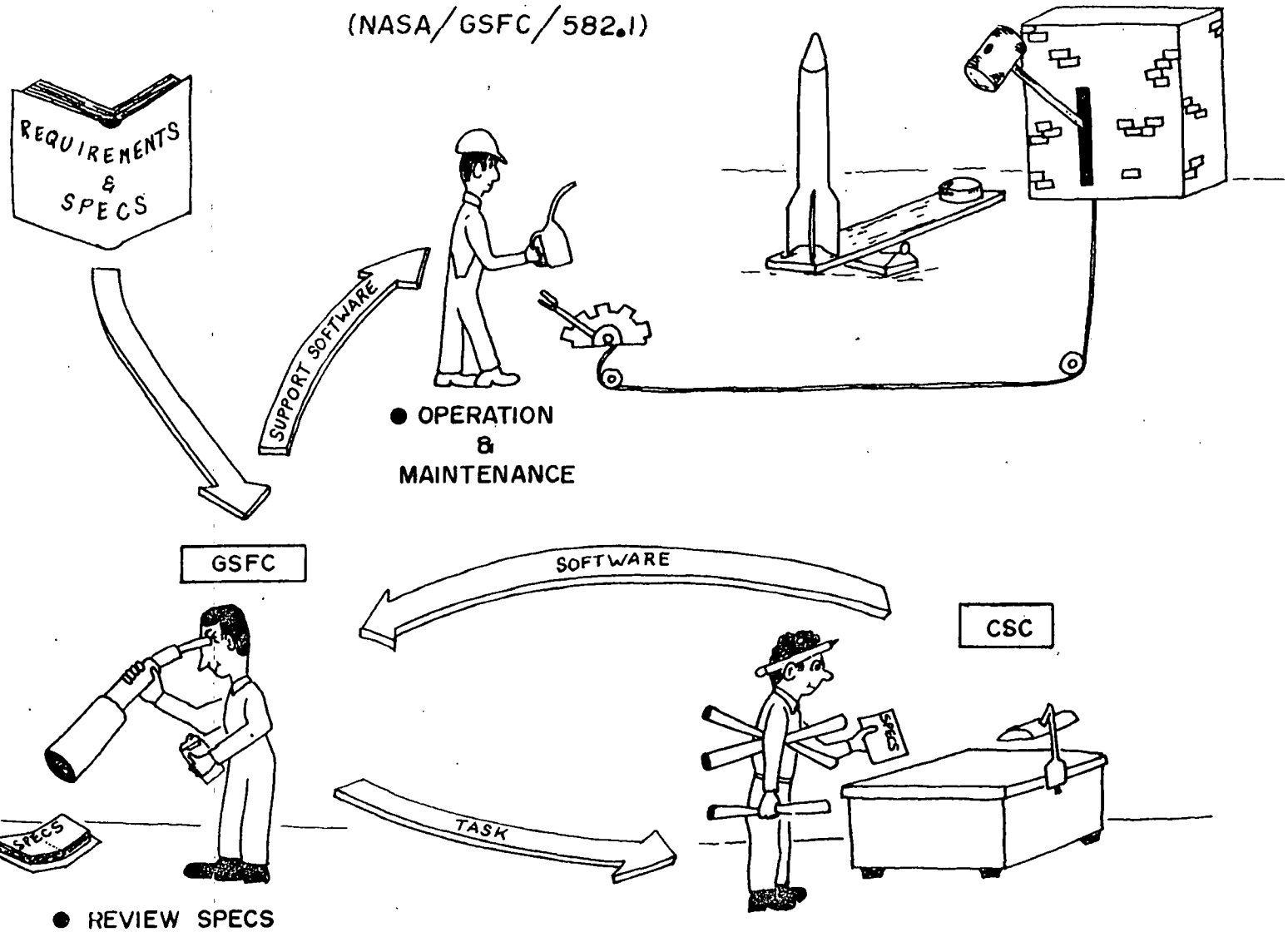
SOFTWARE ENGINEERING LABORATORY



F. McGarry
GSFC

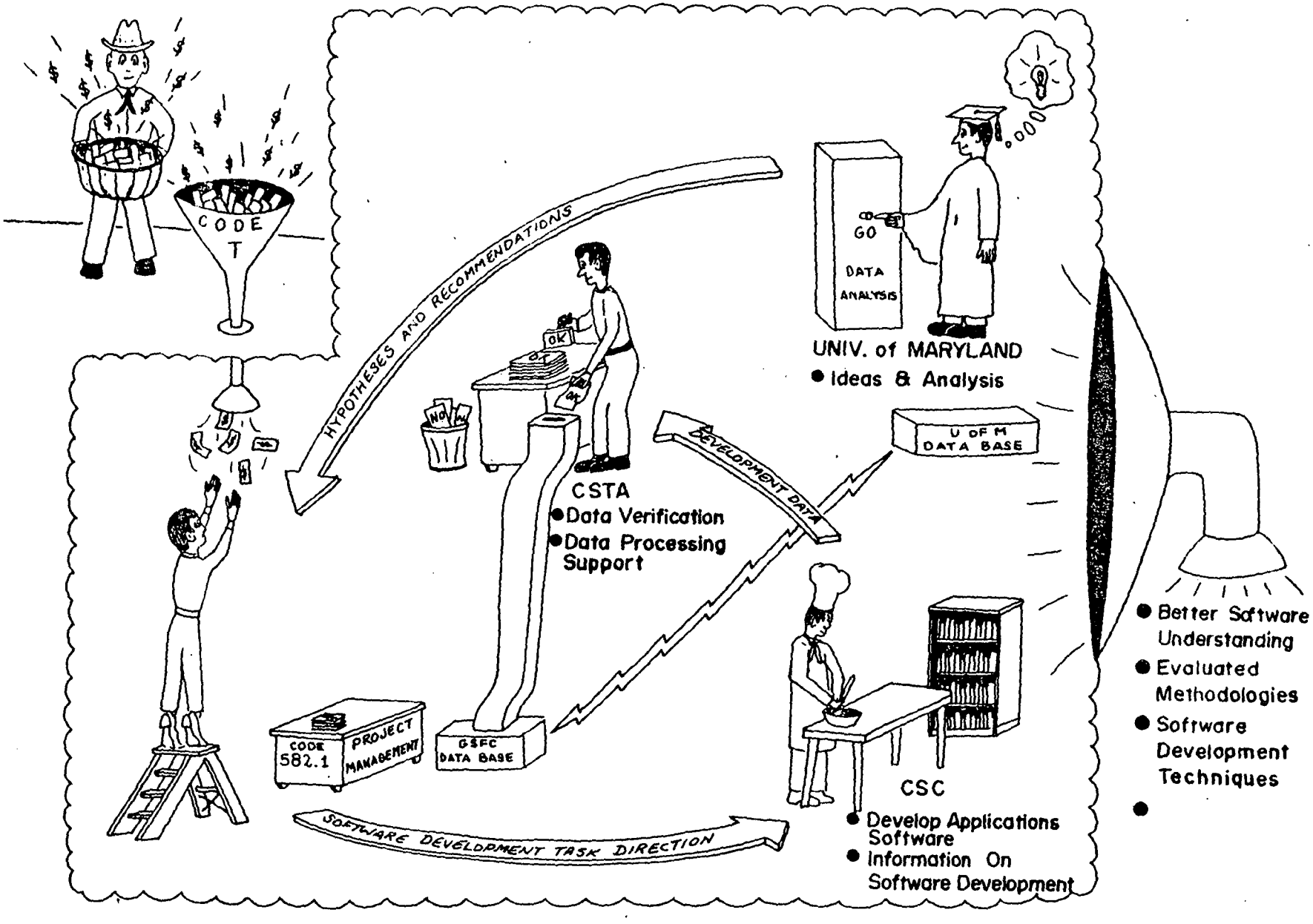
APPLICATIONS SOFTWARE DEVELOPMENT CYCLE

(NASA/GSFC/582.1)



F. McGarry
GSFC
4 of 14

STRUCTURE OF THE SEL



F. McGarry
GSFC
5 of 14

SOFTWARE ENGINEERING LABORATORY – OBJECTIVES

1. UNDERSTAND
   - OUR CURRENT SOFTWARE DEVELOPMENT PROCESS
   - STRENGTHS AND WEAKNESSES
   - TYPE OF ERRORS
   - HOW DO WE SPEND TIME AND MONEY
2. EVALUATE
   - METHODOLOGIES
   - TOOLS
   - MODELS

   } "REAL WORLD" ENVIRONMENT
3. PRODUCE MODEL
   - FOR SOFTWARE DEVELOPMENT
4. IDENTIFY AND APPLY
   - IMPROVED TECHNIQUES

SOFTWARE ENGINEERING LABORATORY – THE PROCESS

- EXPERIMENTS
  - SCREENING (NO PERTURBATIONS)
  - SEMI-CONTROLLED (SPECIFIC METHODOLOGIES APPLIED)
  - CONTROLLED* (TASKS DUPLICATED)
- IDENTIFY INFORMATION REQUIRED
  - FORMS
  - INTERVIEWS
  - AUTOMATIC ACCOUNTING
  - CODE AUDITORS
  - TOOLS (PAN. VALET, . . .)
- ANALYSIS
  - PROFILE INFORMATION
  - APPLY METRICS-MEASURES
  - SOFTWARE MODELING
  - TOOL EVALUATION

F. McGarry
GSFC

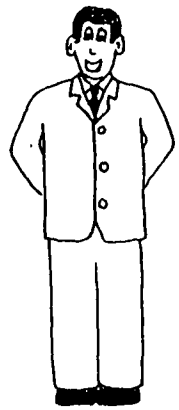
TYPES OF EXPERIMENTS

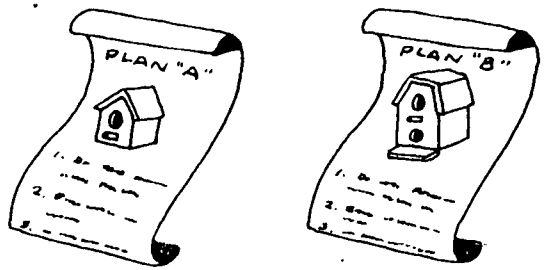
SCREENING TASKS

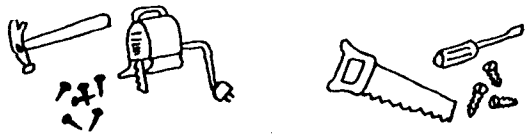
"LET ME WATCH YOU BUILD THE HOUSE."

SEMI-CONTROLLED TASKS

"PLEASE BUILD THE HOUSES AS NOTED IN THE INSTRUCTIONS."

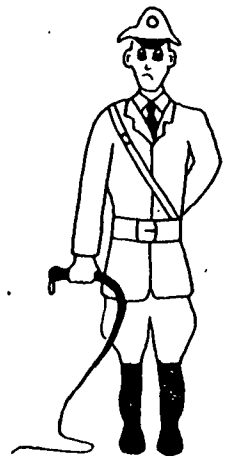


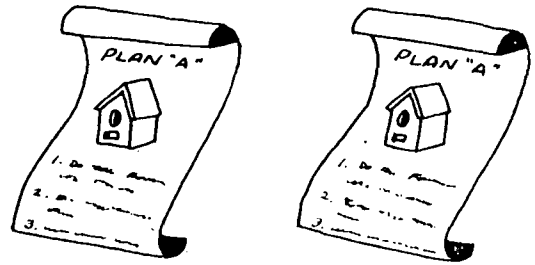


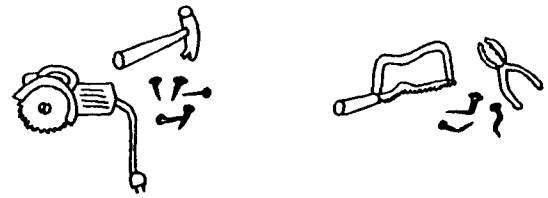


CONTROLLED EXPERIMENT

"YOU WILL BUILD THE HOUSES AS YOU ARE INSTRUCTED."

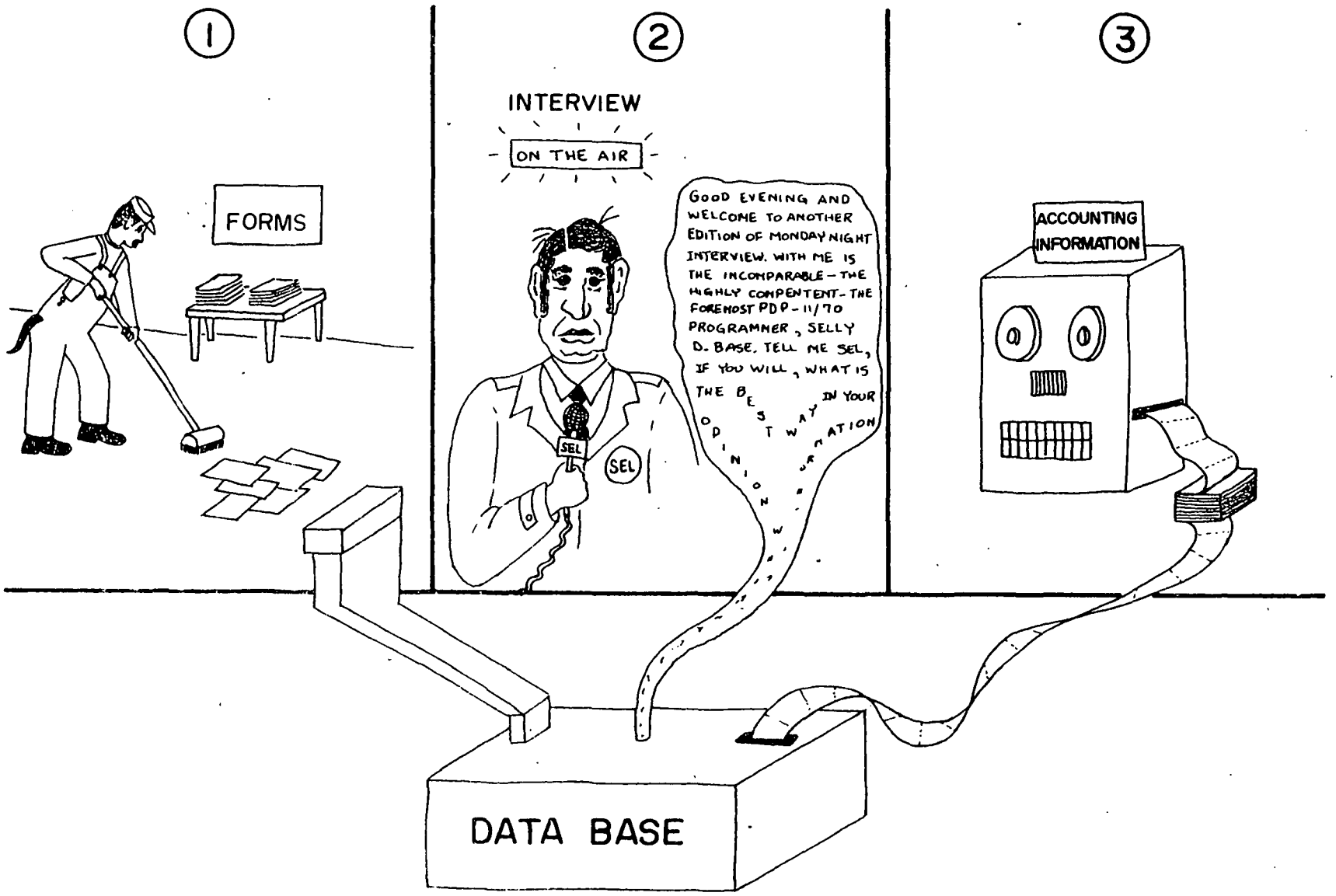






F. McGarry
GSFC
8 of 14

MONITORING THE SOFTWARE DEVELOPMENT PROCESS



SOFTWARE ENGINEERING LABORATORY
DATA COLLECTED

PROJECT	DELIVERED LINES OF SOURCE(K)	TEAM SIZE	DATA COLLECTED METHODOLOGY	COMPLETENESS OF DATA OBTAINED
1	56	7	3-6-7	***
2	3	2	5-6-7	*
3	7	2	2-4-6	**
4	2	2	6	*
5	3	1	2-3	**
6	3	2	3-4	**
7	20	3	2	*
8	62	8	5-6-7	*
9	54	11	1	***
10	70	5	3-4-5-6-7	***
11	3	1	2	***
12	88	11	1-3-4-5-8	***
13	6	2	6	*
14	23	4	6	*
15	6	2	2	*
16	16	2	2	**
17	114	8	1-2-5-6-7	***
18	7	2	6-8	*
19	58	8	1-3-8	***
20	102	10	3-4-5-6-7-8	***

TYPE SOFTWARE

1. SCIENTIFIC
2. UTILITY
3. DATA PROCESSING
4. REAL TIME
5. GRAPHICS

* Assembler Language (All Others FORTRAN)

METHODOLOGY

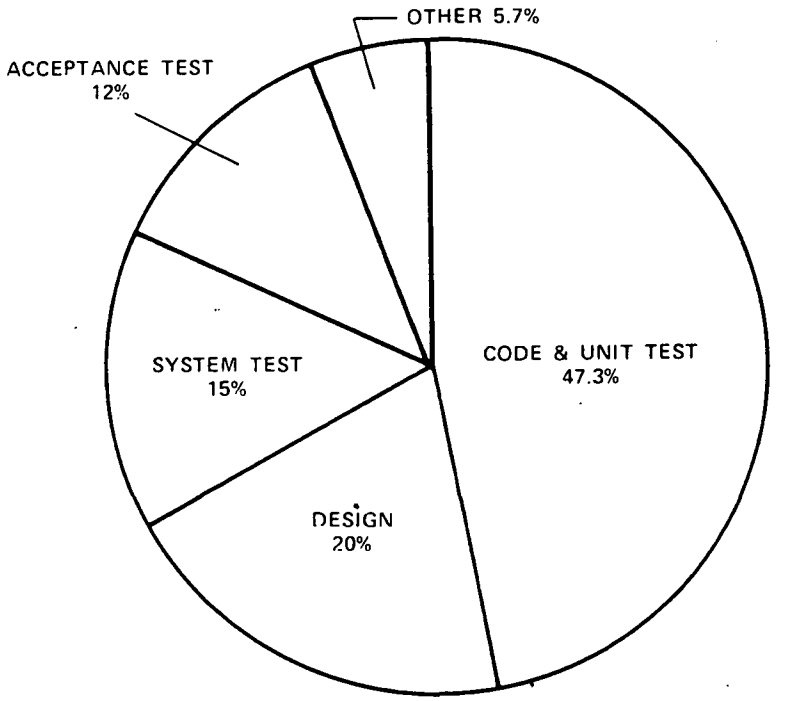
1. CHIEF PROGRAMMER
2. TOP DOWN
3. PRE-COMPILE-STRUCTURE
4. PDL
5. WALK THROUGHS
6. CODE READING
7. LIBRARIAN
8. FORMAL TEST PLAN (DURING DEVELOPMENT)

EXTRACTED DATA

* SOME GOOD DATA
** GOOD DATA
*** VERY GOOD DATA

PROFILE DATA

DISTRIBUTION OF EFFORT BY PHASE



SOURCE: NASA/GSG GSFC (SEL)
AVERAGED 6 PROJECTS (RESOURCE SUMMARY)

PROFILE DATA
EFFORT BY PHASE
(PERCENT)

	TRW	IBM	NASA/GSFC (6 PROJECTS)	NASA/GSFC 1 STUDY TASK COMPONENT STATUS	NASA/GSFC 1 STUDY TASK RESOURCE
CODE	20	30	47	34	50
DESIGN	40	35	20	32	19
CHECKED & TEST	40	25	27	26	19
OTHER		10	6	8	12

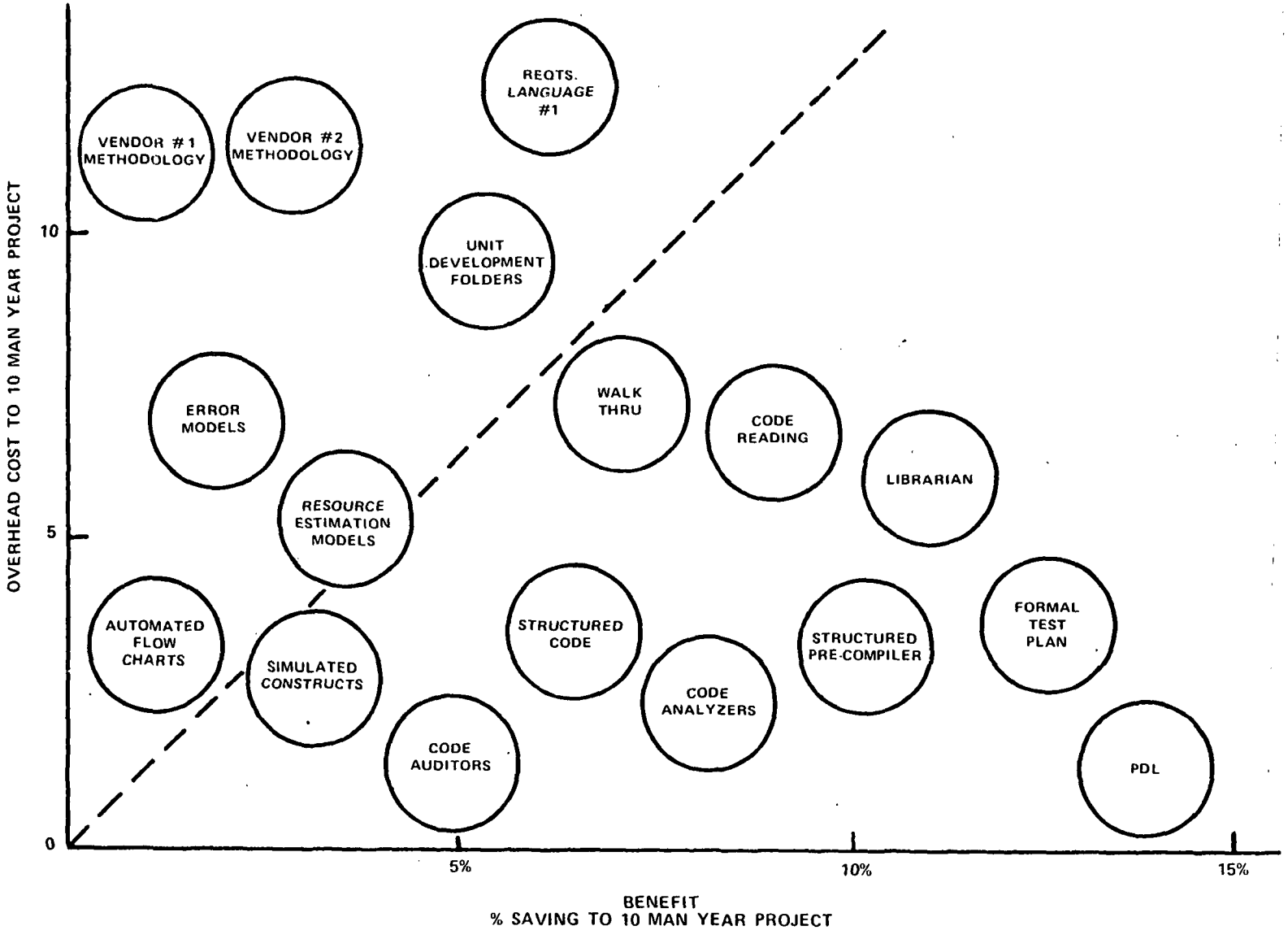
SOFTWARE ENGINEERING LABORATORY

PROJECT NUMBER	(NEW CODE) LINES/ MM	(+20% OLD) LINES/ MM	(NEW CODE) 95 TIME/ 100 LINES	(NEW CODE) 75 TIME/ 100 LINES	(NEW CODE) RUNS/ 100 LINES	% MANAGEMENT	METHODOLOGY FOLLOWED	RESULTS
1	511	512	8.0	24.8	14.9	22.6%	1-3	VERY LATE DELIVERY; EXCEEDED BUDGET; MANY LATE ANOMALIES
2	448	512	6.2	19.6	14.4	14.0%	3	LATE DELIVERY; EXCEEDED BUDGET; MANY LATE ANOMALIES
3	543	546	9.5	20.5	10.5	26.8%	1 8 4 9	ON TIME DELIVERY; WITHIN BUDGET; SOME LATE ANOMALIES
4	715	765	7.4	11.2	14.2	14.5%	3 6 7 9 8 10	ON TIME DELIVERY; WITHIN BUDGET; SMOOTH FINISH
5	504	755	12.7	17.8	13.9	28.7%	1 4 2 5 7 6 8 9	EARLY DELIVERY; WELL WITHIN BUDGET; NO LATE ANOMALIES

TOOLS AND METHODOLOGIES

1 . . . STRUCTURED PRE-COMPILER
2 . . . PDL
3 . . . CHIEF PROGRAMMER
4 . . . UNIT DEVELOPMENT FOLDERS
5 . . . FORMAL TRAINING IN S.E. TECHNIQUES
6 . . . FORMAL TEST PLAN
7 . . . WALK THROUGH
8 . . . CODE READING
9 . . . LIBRARIAN
10 . . . TOP DOWN

SOFTWARE METHODOLOGIES
SUBJECTIVE EXPERIENCE



CONCLUSION

- DATA COLLECTION IS IMPORTANT
- UNDERSTAND THE LOCAL ENVIRONMENT
- COST ABSORBED IN BENEFITS
- THERE ARE MODELS THAT DESCRIBE OUR SOFTWARE ENVIRONMENT
- SOFTWARE TOOL AND METHODOLOGIES DO EFFECT THE SOFTWARE DEVELOPMENT PROCESS
- THE SOFTWARE DEVELOPMENT PROCESS CAN BE IMPROVED
- THERE ARE METRIC THAT DO MEASURE THE "GOODNESS" OF SOFTWARE