SOFTWARE MANAGEMENT ENVIRONMENT FOR NASA

Frank E. McGarry

SOFTWARE MANAGEMENT ENVIRONMENT FOR NASA

OBJECTIVE

Develop. assess and implement software management aids
(tools, measures, techniques)
leading to an environment producing software of 'increased quality'
(reliability and life cycle cost)

AREAS OF INVESTIGATION

Design and specification measures
Management tools (including rapid prototyping aids)

SOFTWARE MANAGEMENT ENVIRONMENT

AREAS OF CONSIDERATION

DESIGN/SPECIFICATION MEASURES

Can we determine 'quality' of design (or specs)?
What is 'quality' for design?
How do we determine trade-offs for various design approaches?
Can I determine early what part of system is 'easy' or 'hard'?

MANAGEMENT TOOLS

Given existing development information, PREDICT-ASSESS-SELECT-CONTROL
Automatically determine quality of design
Automatically determine 'improved' design
Evaluate specs

NASA

SOFTWARE ENGINEERING LABORATORY DATA STUDIED

TYPE OF SOFTWARE: SCIENTIFIC, GROUND-BASED, INTERACTIVE GRAPHIC, MODERATE RELIABILITY AND RESPONSE REQUIREMENTS

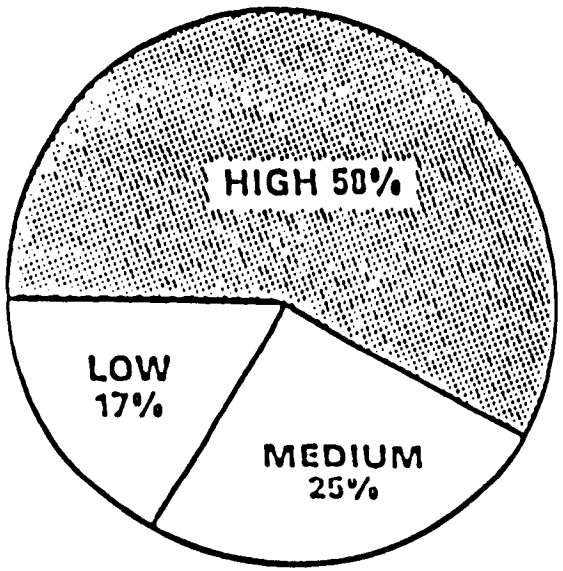
LANGUAGES: 85% FORTRAN, 15% ASSEMBLER MACROS

MACHINES: IBM S/360 AND 4341, BATCH WITH TSO

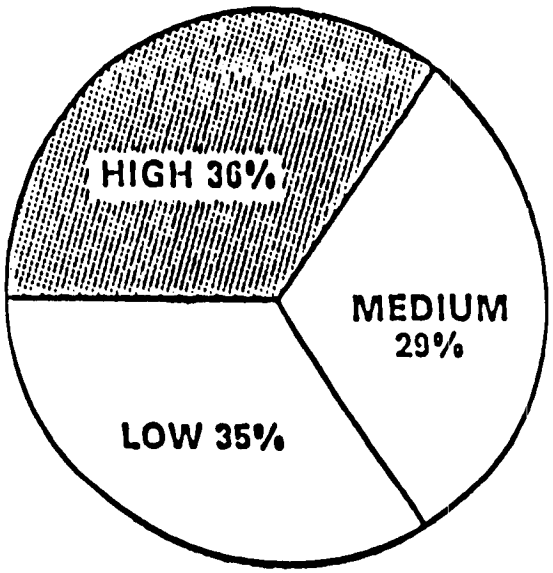
PROJECT CHARACTERISTICS:	AVERAGE	HIGH	LOW
DURATION (MONTHS)	15.6	20.5	12.9
EFFORT (STAFF-YEARS)	8.0	11.5	2.4
SIZE (1000 LOC)			
DEVELOPED	57.0	111.3	21.5
DELIVERED	62.0	112.0	32.8
STAFF (FULL-TIME EQUIV.)			
AVERAGE	5.4	6.0	1.9
PEAK	10.0	13.9	3.8
INDIVIDUALS	14	17	7
APPLICATION EXPERIENCE			
MANAGERS	5.8	6.5	5.0
TECHNICAL STAFF	4.0	5.0	2.9
OVERALL EXPERIENCE			
MANAGERS	10.0	14.0	8.4
TECHNICAL STAFF	8.5	11.0	7.0

SAMPLE: 22 SYSTEMS USING A VARIETY OF TECHNOLOGIES

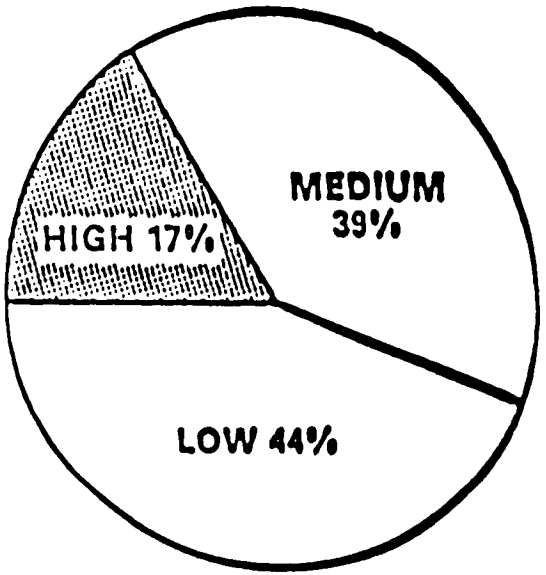
STRENGTH AS A DESIGN MEASURE



HIGH STRENGTH



MEDIUM STRENGTH



LOW STRENGTH

HIGH STRENGTH IMPLIES HIGH RELIABILITY

BASED ON: * 480 Modules

* 3 Projects

RELIABILITY:

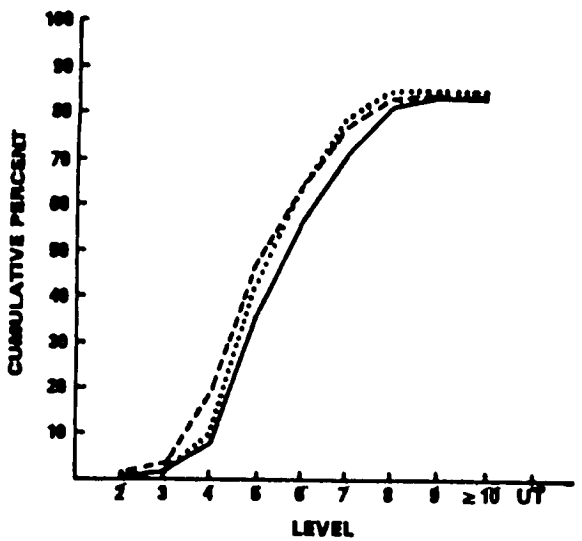
High Error=0/1000 L.O.C.

Med Error.LE.2/1000 L.O.C.

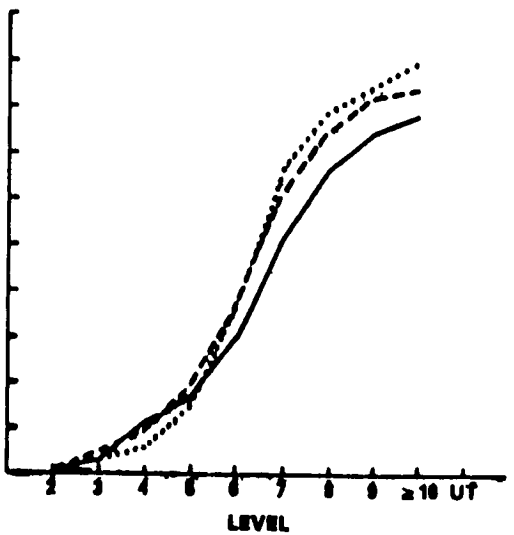
Low Error.GT.2/1000 L.O.C.

DESIGN IS A PARTITIONING OF STRUCTURE

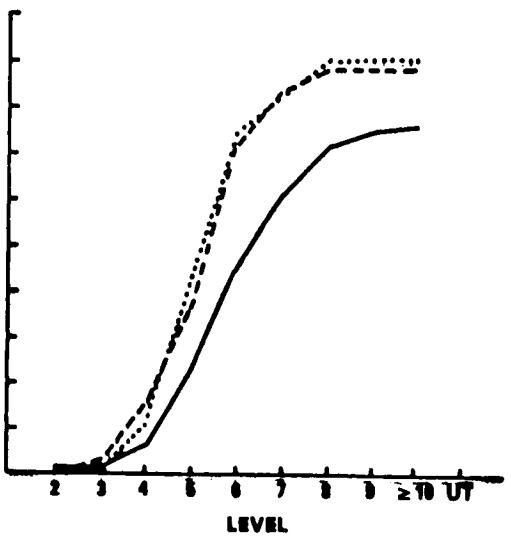
PROJECT 1: GOOD



PROJECT 2: MEDIUM



PROJECT 3: POOR



KEY:

- CONTROL STRUCTURE (FAN-OUT)
- DATA STRUCTURE (VARIABLES)
- SOFTWARE STRUCTURE (MODULES)

Developing 'Specification' Measures

OUR APPROACH

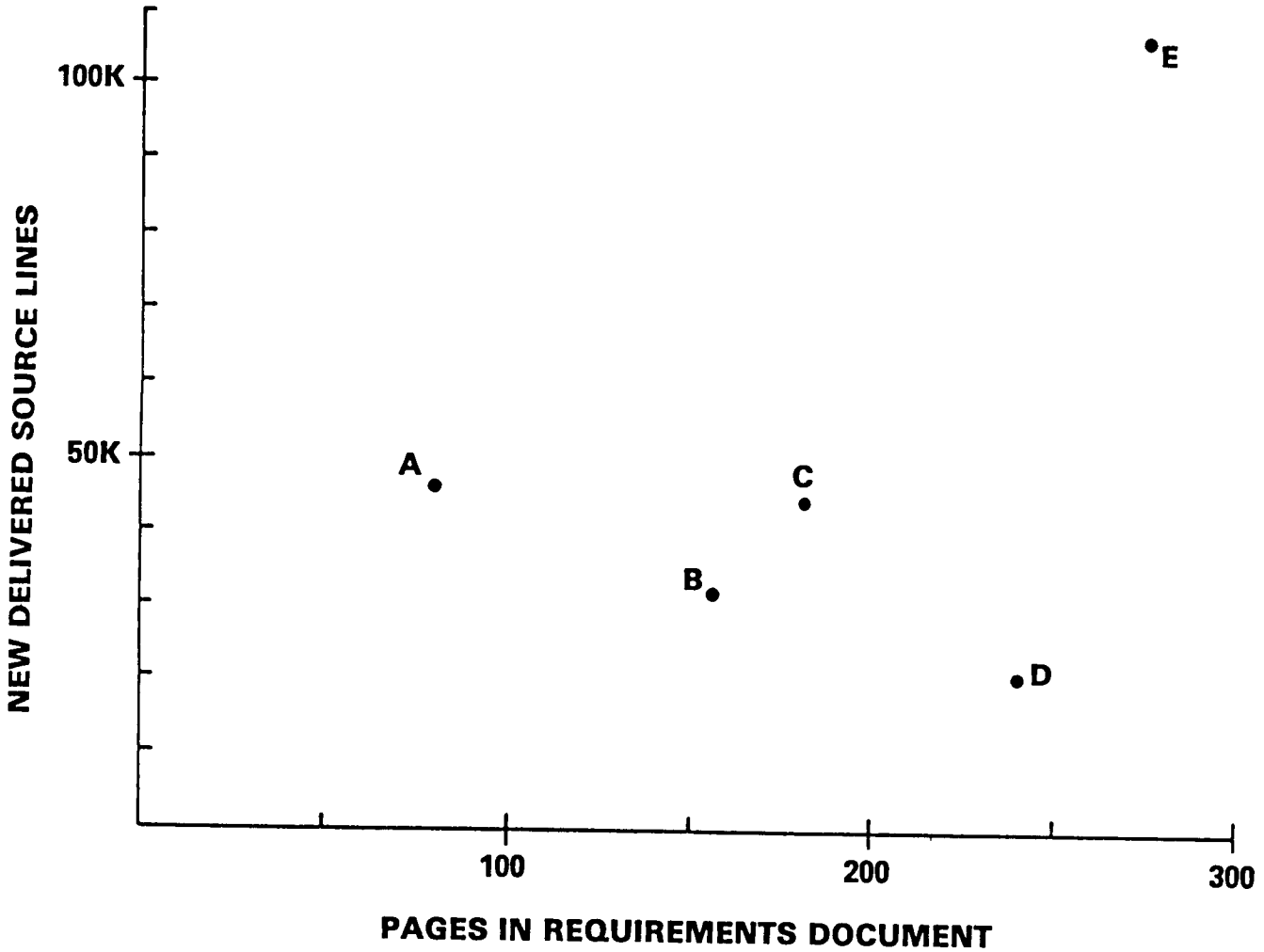
FOCUS: OBJECTIVE MEASURES

PROCEDURE: DEFINED 29 EXPLICIT MEASURES BASED ON EXISTING REQUIREMENTS SPECIFICATIONS

NUMBER OF PAGES
NUMBER OF CONSTRAINTS
NUMBER OF I/O REQUIREMENTS
o
o
o

RESULT: MEASURES WERE EXTRACTABLE BUT NOT USEFUL

FIVE FLIGHT DYNAMICS SOFTWARE PROJECTS
NEW SOURCE LINES VS. PAGES OF REQUIREMENTS



LESSON: TO DEVELOP OBJECTIVE SPECIFICATION MEASURES, REPRESENTATION IS EVERYTHING!

OUR REVISED APPROACH

STEP 1: PROPOSE A NEW REPRESENTATION

STEP 2: DEFINE SPECIFICATION MEASURES BASED ON IT

STEP 3: APPLY IT TO A REAL SYSTEM

STEP 4: EXTRACT THE MEASURES

STEP 5: ASSESS THE PROCESS AND THE RESULTING MEASURES

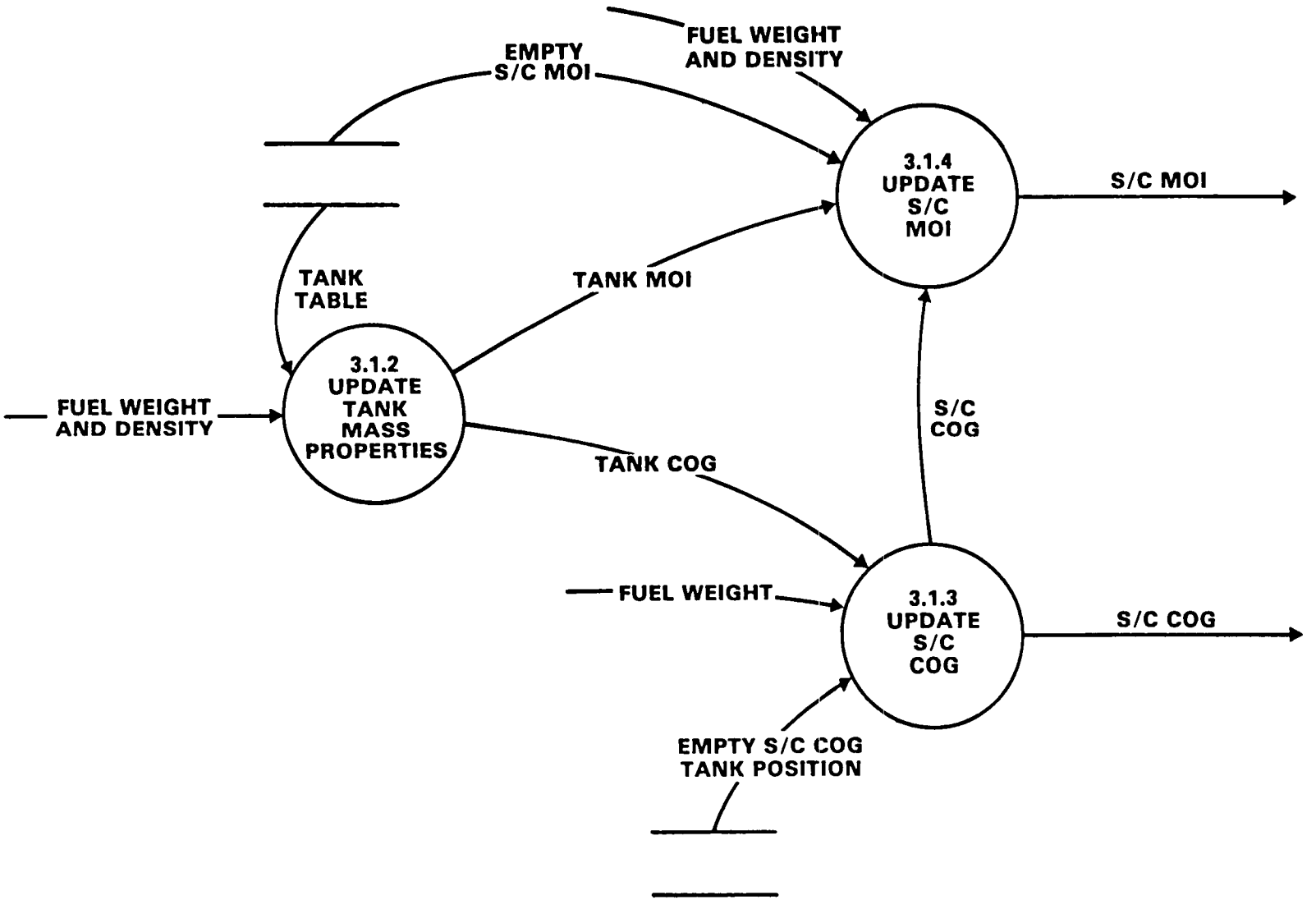
STEP 1:
PROPOSE A NEW REPRESENTATION

COMPOSITE SPECIFICATION MODEL (CSM)

RATIONALE: REQUIREMENTS FOR COMPLEX SOFTWARE NEED TO BE SPECIFIED FROM MULTIPLE VIEWPOINTS

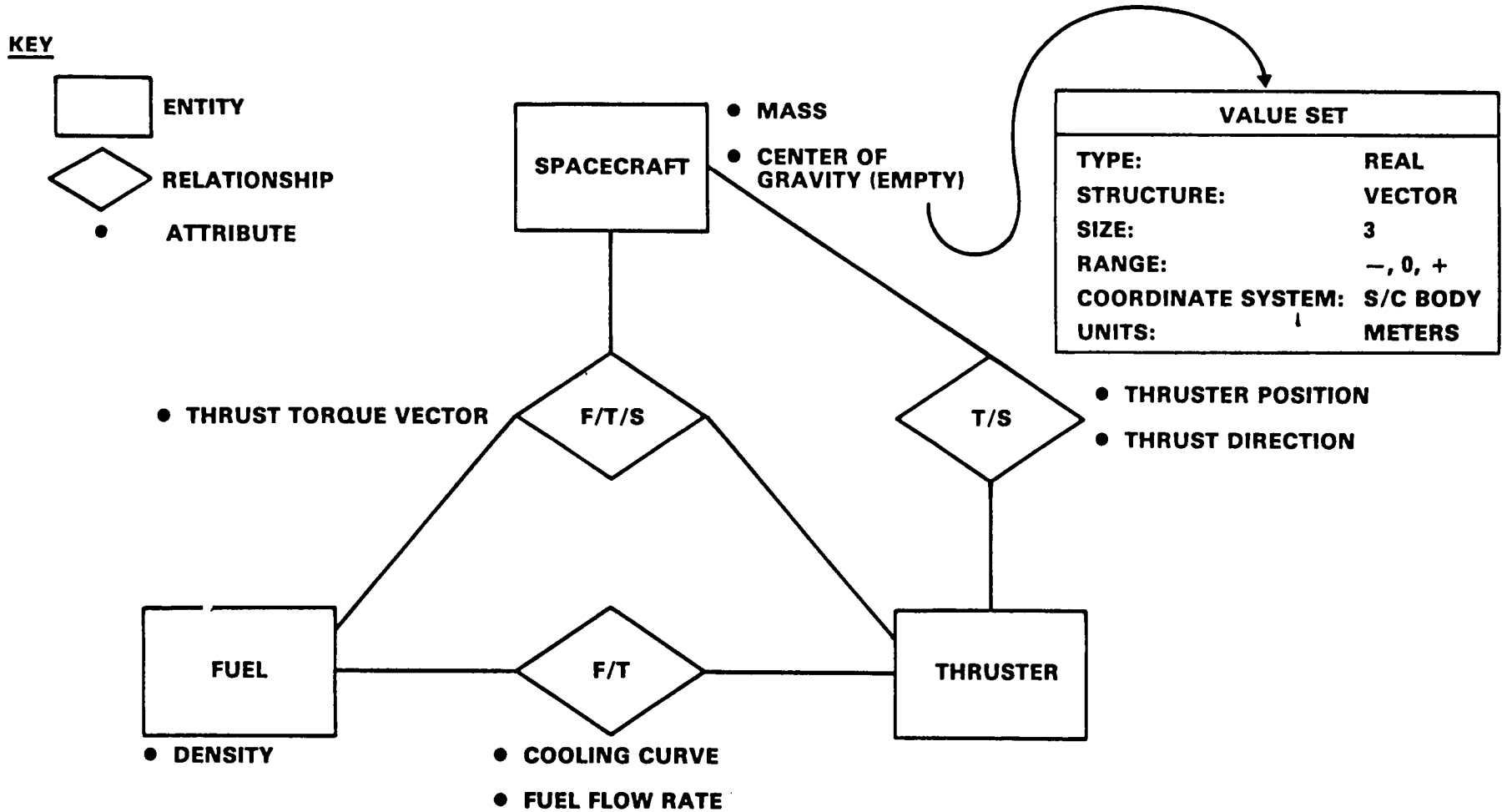
VIEWPOINT	NOTATION
• FUNCTIONAL	• DATA FLOW
• CONTEXTUAL	• ENTITY/RELATIONSHIP
• DYNAMIC	• STATE/TRANSITION

EXAMPLE OF FUNCTIONAL VIEW

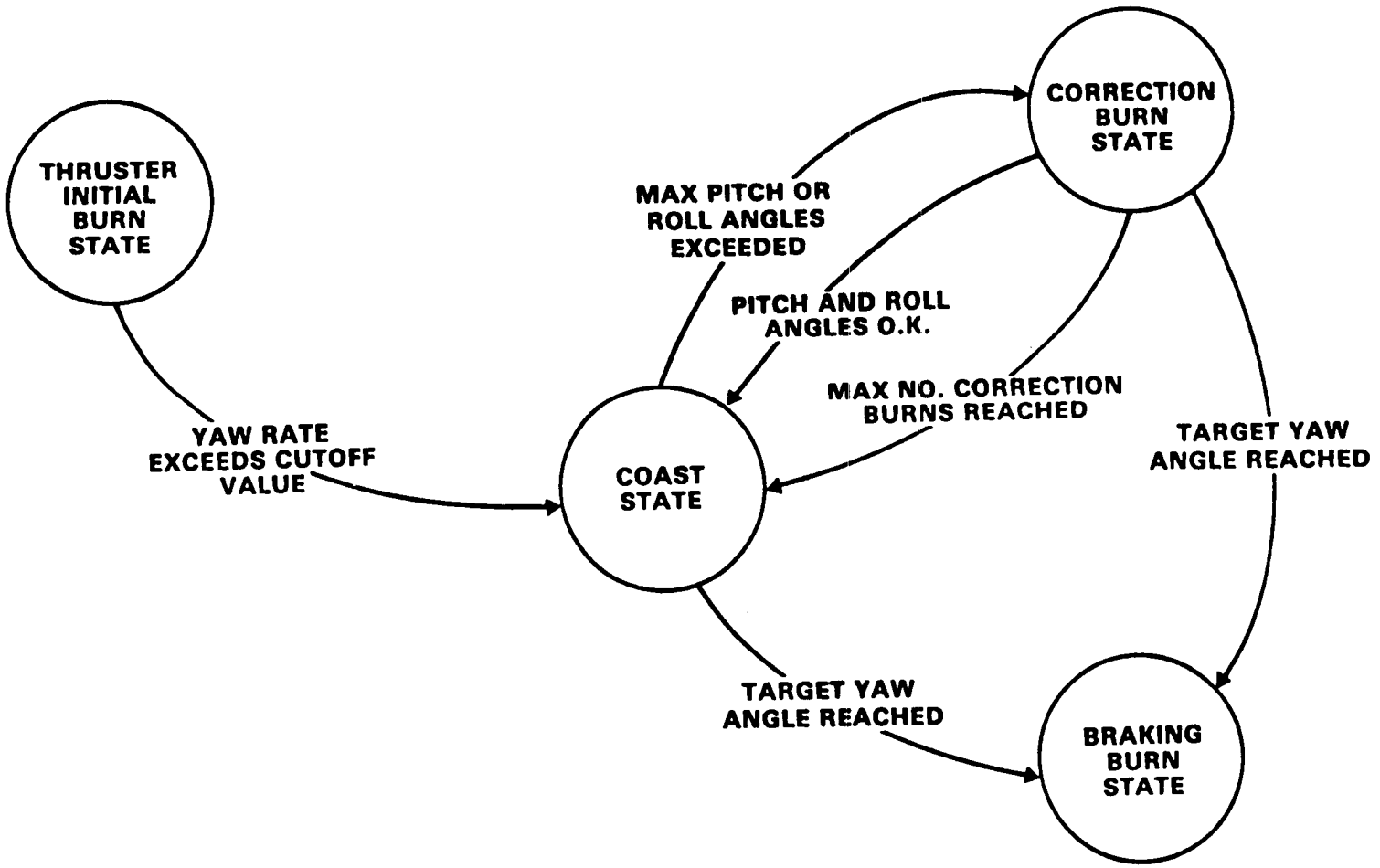


714-AGR-(89c)

EXAMPLE OF CONTEXTUAL VIEW



EXAMPLE OF DYNAMIC VIEW (STATES AND TRANSITIONS)



714-AGR-(88a)

STEP 2: DEFINE MEASURES BASED ON THE COMPOSITE SPECIFICATION MODEL

58 MEASURES DEFINED

EXPLICIT	ANALYTIC
NUMBER OF FUNCTIONAL PRIMITIVES	WEIGHTED FUNCTION
NUMBER OF DATA ITEMS	RELATION DENSITY
NUMBER OF STATES	ARC WEIGHT
●	●
●	●
●	●

STEP 3: APPLY THE COMPOSITE SPECIFICATION MODEL TO A REAL SYSTEM

- YAW MANEUVER CONTROL UTILITY OF EARTH RADIATION BUDGET SATELLITE (ERBS)
- FORTRAN
- 11,200 DELIVERED SOURCE LINES
- 85 MODULES

714-AGR-(89a)

STEP 4 EXTRACT THE MEASURES

MEASURE	VALUE
FUNCTIONAL VIEW	
• FUNCTIONAL PRIMITIVES	39
• INTERFACE COUNT	3
• INTERNAL ARCS	60
• INTERNAL DATA ITEMS	42
• SYSTEM IN/OUT DATA ITEMS	67
• FILE IN/OUT DATA ITEMS	74
• WEIGHTED FUNCTION	688
CONTEXTUAL VIEW	
• ENTITIES	11
• EVENTS	14
• RELATIONS	19
• ATTRIBUTES	91
• VALUE SETS	29
DYNAMIC VIEW	
• STATES	7
• TRANSITIONS	11

STEP 5: ASSESS THE PROCESS AND RESULTING MEASURES

PROCESS

- EFFORT REQUIRED FOR CSM MAY REDUCE EFFORT IN LATER PHASES
  - 2.1 STAFF MONTHS FOR TRADITIONAL REQUIREMENTS ANALYSIS
  - 1.7 STAFF MONTHS FOR BUILDING CSM

RESULTING MEASURES

- HUMAN JUDGMENT STILL IS A FACTOR
- NEED TO MEASURE MORE PROJECTS

CONCLUSIONS

- OBJECTIVE SPECIFICATION MEASURES NEED DISCIPLINED REPRESENTATION OF REQUIREMENTS
- BUILDING THE CSM IS FEASIBLE
  - YIELDS OBJECTIVE SPECIFICATION MEASURES
  - MULTIPLE VIEWS ARE MORE REVEALING
  - MORE EFFECTIVE REPRESENTATION TO BEGIN DESIGN
- CAPTURING THE CONTEXT OF A SYSTEM IS BENEFICIAL
  - SOURCE OF CHANGES TO THE SYSTEM
  - LOGICAL PREDECESSOR OF OBJECT-ORIENTED DESIGN

DYNAmic Management Information Tool

The Idea

INPUT

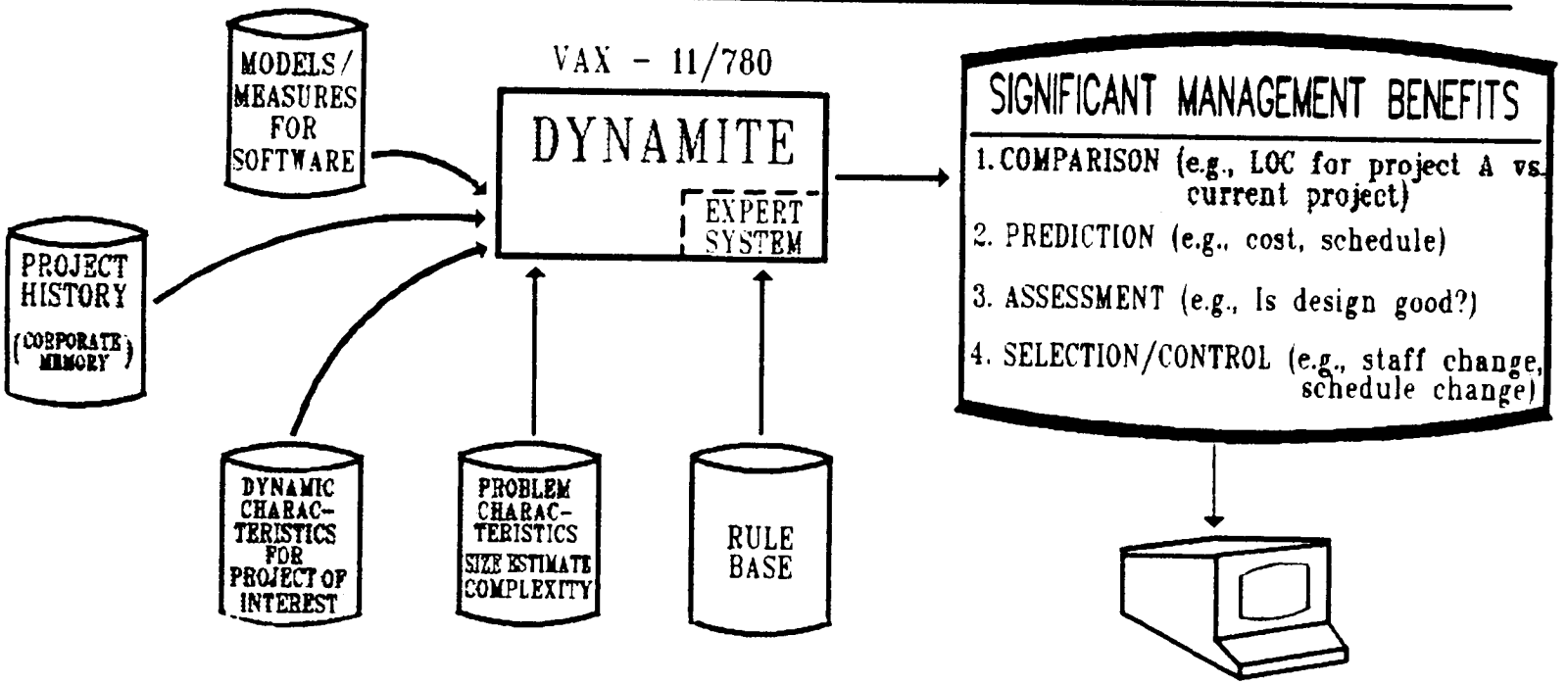
1. Verified Measures/models for Development
   (e.g. 40–20–40 Rule or Rayleigh Curve)
2. Past Project Histories
   (e.g. Staffing Profiles)
3. Verified 'RULES' of Software Development
   (e.g. If excessive ECR's
   then specs are of poor quality)
4. Current Project Development Data
   (e.g. Staffing, Changes, Resource Consumption)

OUTPUT

1. PREDICT
   (e.g. When will project be complete?)
2. ASSESS
   (e.g. Tesing procedures are bad)
3. COMPARE
   (e.g. Relative to past projects, the code development rate is very low.)
4. SELECT/CONTROL
   (e.g. Use tighter testing standards for this project.)

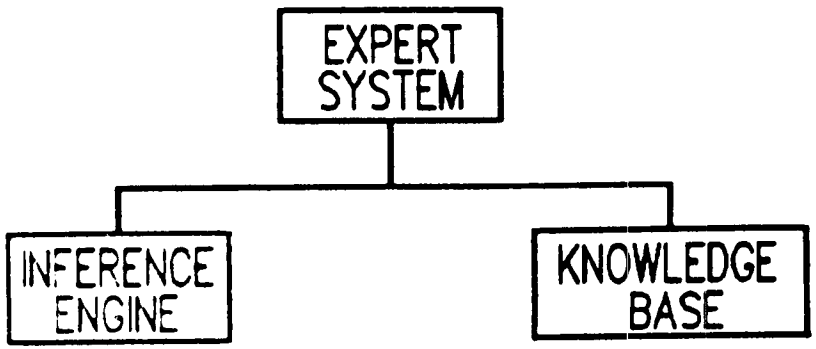
SOFTWARE MANAGEMENT ENVIRONMENT

DYNAMIC MANAGEMENT INFORMATION TOOL (DYNAMITE)



* SHADED = FY84 capabilities
* OPEN = Planned Work

DYNAMITE EXPERT SYSTEM

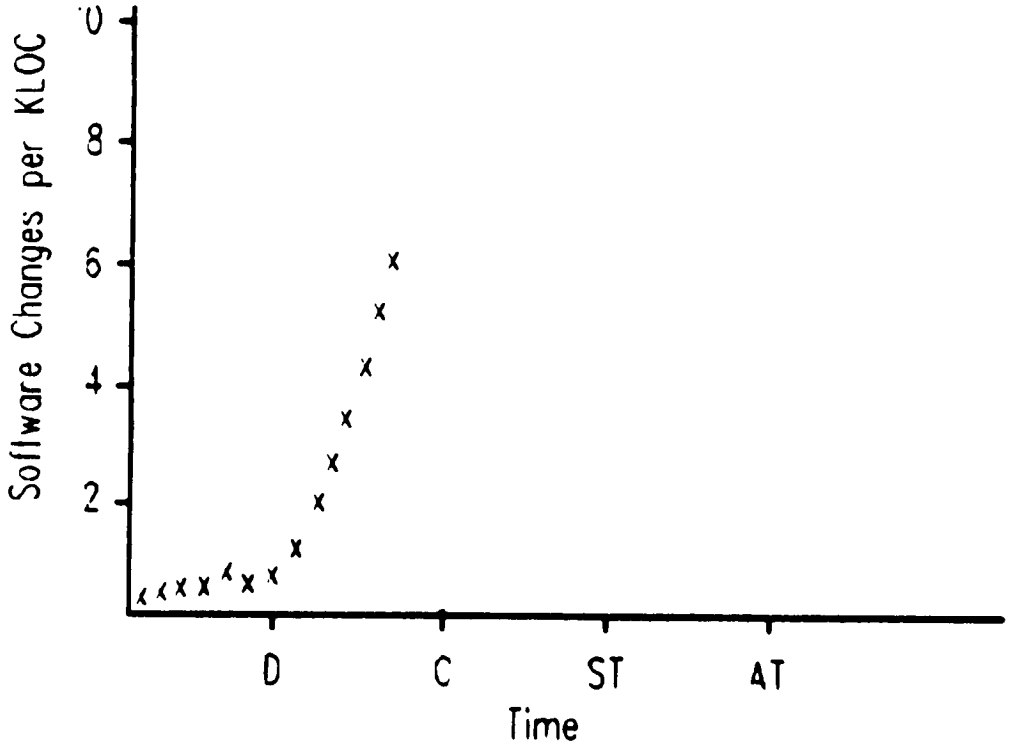


University of
Maryland Tool
KMS - Knowledge
Management System

3 representations of knowledge
- Rule based
  approximately 100 rules
- Frame based
- Hypothesize and Test

DYNAMITE SCENARIO UTILIZING EXPERT SYSTEM

STEP 1
Retrieve data
from Dynamic
project file



SAMPLE RULES

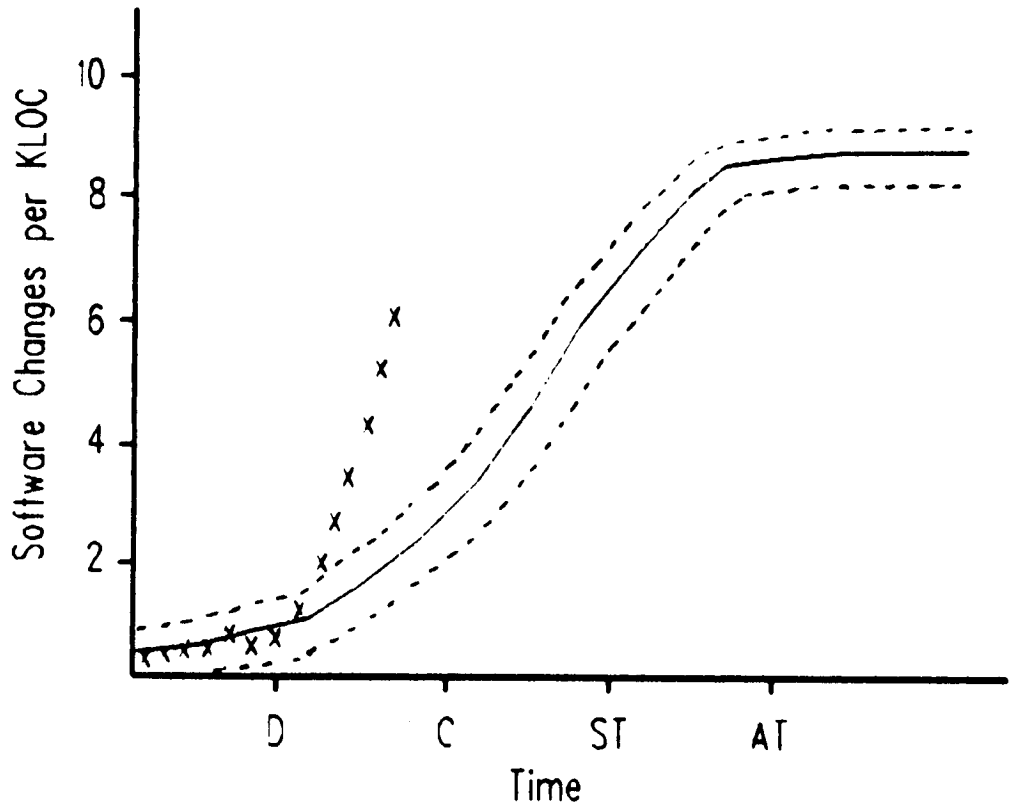
RULE 1: If computer run per line of source code is above normal
and in early code phase then
interpretation is
lots of testing 75%
error-prone code 75%
high complexity or tough problem 50%
low productivity 25%
removal of code by testing or transporting 25%

RULE 2: If software changes per line of source code is above normal
and in system test phase then
interpretation is
error-prone code 75%
unstable specification 75%
loose configuration management or unstructured development 75%
good testing or good test plan 25%
removal of code by testing or transporting 25%
near build or milestone date 25%

DYNAMITE SCENARIO UTILIZING EXPERT SYSTEM

STEP 1
Retrieve data
from Dynamic
project file

STEP 2
Retrieve Mode
of SEL Experience

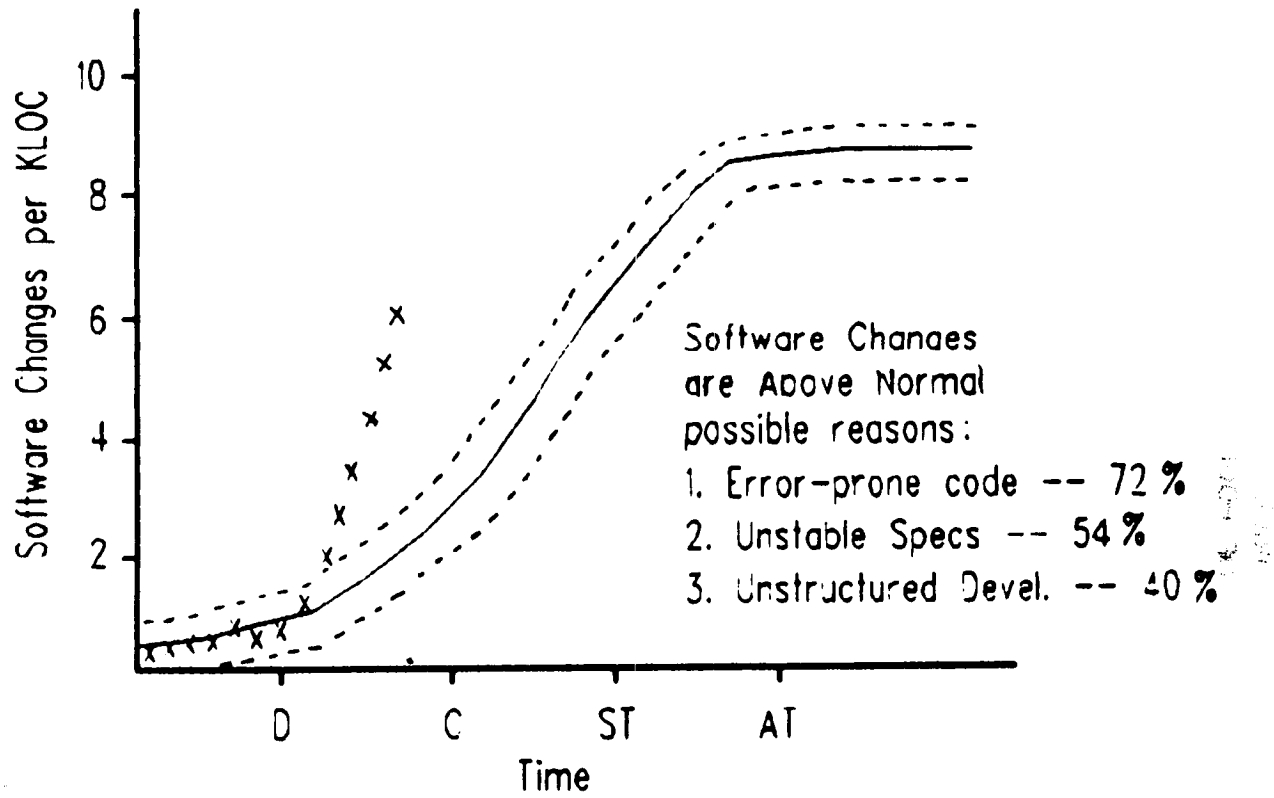


DYNAMITE SCENARIO UTILIZING EXPERT SYSTEM

STEP 1
Retrieve data from Dynamic project file

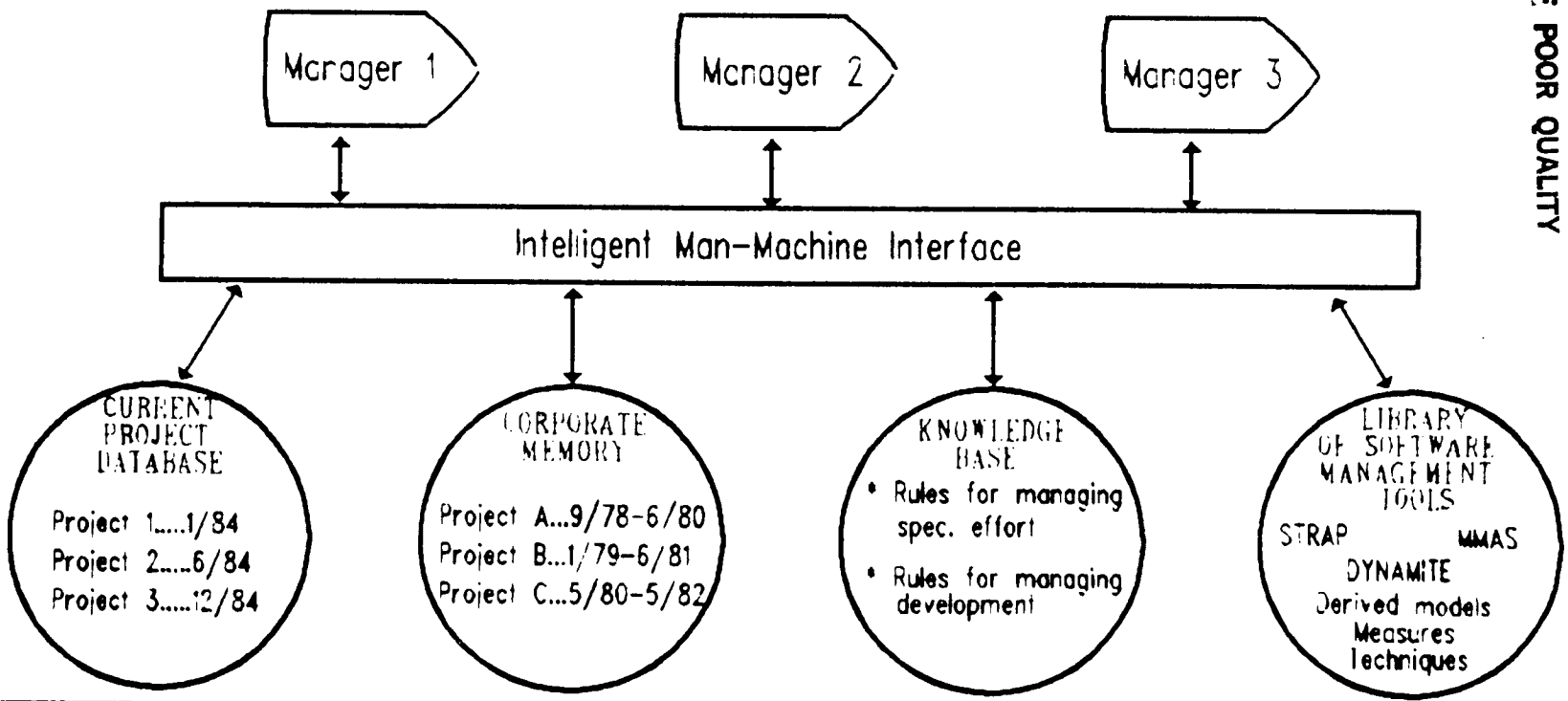
STEP 2
Retrieve Mode of SEL Experience

STEP 3
Assess meaning of Comparison



SOFTWARE MANAGEMENT ENVIRONMENT

Functional Diagram



NASA