A COMPREHENSIVE PLANNING & CONTROL TOOL FOR LARGE SOFTWARE PROJECTS

R.A. Doering
C-17 Chief Engineer

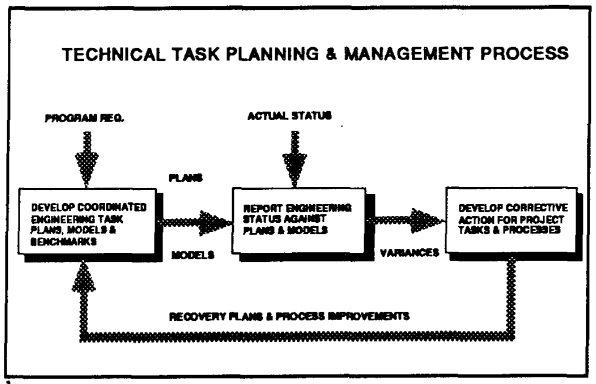
Delco Electronics Corporation
Delco Systems Operations
6767 Hollister Avenue
Goleta, CA 93117

Delco Systems Operations and Douglas Aircraft Company have been developing the over 300,000 lines of Jovial software for Delco's Mission Computer to be installed on the C-17 Aircraft. A comprehensive system has been developed to plan, control and report on the software development progress for this large effort.[1]

A flexible relational data base manager is being used to track scheduled and actual accomplishment twenty three software design, code and test activities or milestones. Over 2700 modules of executable code and data statements distributed over up to three simultaneous release versions are monitored. Various reports, metrics and indicators are described. Emphasis is placed on the teamwork and cultural changes that are essential to the successful implementation of the method.

INTRODUCTION

The award to Delco Systems Operations of an avionics systems contract, which included a large scale software development activity, acted as a catalyst to initiate a more disciplined approach to software project measurement and control. The software project was the operational flight program for the Mission Computer of the C-17 military airlifter. Hundreds of software and systems engineers working over a five year period developed approximately 300,000 lines of Jovial code in a military specification environment. The need for project status visibility, flexible planning and cost control, convinced us to initiate a comprehensive software process measurement and management system. The system described here has evolved and been institutionalized over the past five years and will continue to be improved and expanded in the future as part of our Total Quality Management (TQM) effort as shown below.


TECHNICAL TASK PLANNING & MANAGEMENT PROCESS
PROGRAM REQ.
ACTUAL STATUS
PLANS
MODELS
VARIANCES
DEVELOP COORDINATED ENGINEERING TASK PLANS, MODELS & BENCHMARKS
REPORT ENGINEERING STATUS AGAINST PLANS & MODELS
DEVELOP CORRECTIVE ACTION FOR PROJECT TASKS & PROCESSES
RECOVERY PLANS & PROCESS IMPROVEMENTS

[1] This work supported by a government contract and had government sponsorship.

The most difficult aspect in this endeavor is the parsimonious quantification of the work scope so that measurements give a fair estimate of task completion without being burdensome or detrimental to the software development team. A model of well defined verifiable milestones, work packages and quality gates for all phases of the software development cycle was developed. This model was tailored to the specific contractual software development requirements as documented in the Computer Program Development Plan and the Software Configuration Management Plan.

MODELS OF VARIOUS PHASES OF SOFTWARE DEVELOPMENT

Based on a lowest common denominator of work scope, and using the guidelines of DOD-STD-2167A (Defense System Software Development), the software development process was divided into the following phases:

Software Phase	Work Unit
Top Level Design	list of specific activities
Implementation	detailed design, code, test of units
Software Integration	test cases for functional threads
Software Qualification	test procedures for all requirements

This paper will focus on the Implementation phase where the attributes and work items associated with each of approximately 2700 software units (modules) are tracked and reported.

DETAILS OF THE IMPLEMENTATION MODEL

To schedule work items and track progress of executable code, we subdivided each phase of implementation (detailed design, coding, and testing) into milestones that are clearly defined and auditable. By giving weight to each milestone, we took subjectivity out of status reporting. When all milestones in each phase are complete, that phase of implementation for that unit is by definition complete. When some milestones are complete, the percentage complete is calculated based on the weight given each milestone.

CH3158-3/92/0000-0587 $1.00 © 1992 IEEE

The Detailed Design phase of implementation is subdivided into seven milestones. Their rudimentary definitions and assigned weights are:

o	Establishment of Software Development Folder (SDF)	10%
o	Allocation of system requirements to software units.	10%
o	Identification of any open requirements issues	5%
o	Identification of unit's criticality	5%
o	Writing of Program Description Language (PDL)	40%
o	Detailed Design Walkthrough	20%
o	Placement of PDL file in Configuration Management (CM) library.	10%

The Coding phase is subdivided into five milestones:

o	Writing of Jovial code	40%
o	Updating of PDL to assure agreement with Jovial code	20%
o	Updating of size estimates	10%
o	Updating of SDF with source code	10%
o	Completion of code review.	20%

Testing is subdivided into five milestones:

o	Writing of test procedures	25%
o	Successful execution of test procedures	35%
o	Updating of SDF	10%
o	Completion of test review	20%
o	Placement of unit source code and test results in software configuration management (SCM) library.	10%

Similarly, five milestones were defined to control the development of units of data statements (compools).

SYNTHESIZING THE PLAN AND UPDATING STATUS

We developed a relational database to manage the multi-faceted collection of information dealing with attributes, schedule and status of individual units. The advantages of using a relational database are flexibility in reporting capabilities and the relative ease with which changes in management's reporting requirements can be addressed by expanding and creating new tables and applications.

Each unit of executable code or data statements was assigned attributes based on its function, design and placement in specific versions of the delivered software. Using each unit's attributes as its

MC OFP ALL SOFTWARE UNITS REQUIRED FOR PLAN WEEK 273

ALL UNITS REQUIRED FOR PLAN WEEK 273
AS OF STATUS WEEK 271

16-Sep-91

										Software Detail Design									Software Unit Code								Software Unit Test				
UNIT	LNG	SUBBLD	Bfnt SLCS	Dulv SLCS	Mem Size	MEM EST MTHD	CPU	Prg	Est Cpl	SDF EST 10%	Reqs to UNIT 10%	CPDS OK 5%	UNIT CRIT 5%	PDL WRIT 40%	DD Wk Thr 20%	PDL in CM 10%	Cd Prg	Est Cpl	CODE WRIT 40%	UPDATE PDL 20%	CD SZ UPDT 10%	CD SDF UPDT 10%	CD RVW Cplt 20%	UT Prg	Est Cpl	UT PR WRIT 25%	UT PR Xctd 35%	UT SDF UPDT 10%	UT REVIEW D 20%	CD-UT in CM 10%
KM00'MANAGE'MONITOR'SKE	JOV	XFT2.1	187	187	420	A	1	XYZ	263	C	C	N	N	C	C	C	ABC	268	C	C	C	C	C	XYZ	*					
KM01'PERFORM'GLOBAL'INPUT	JOV	XFT2.1	436	436	1457	A	1	XYZ	264	C	C	N	N	C	C	C	ABC	268	C	C	C	C	C	XYZ	*					
KM02'TURN'OFF'ANNUN'MSG	JOV	XFT2.1	40	40	87	A	1	XYZ	263	C	C	N	N	C	C	C	ABC	268	C	C	C	C	C	XYZ	*	C	C	C	C	
KM03'INVALIDATE'OUTPUT	JOV	XFT2.1	91	91	207	A	1	XYZ	263	C	C	N	N	C	C	C	ABC	268	C	C	C	C	C	XYZ	*	C	C	C	C	
KM04'PERFORM'SKE'VALID'CHK	JOV	XFT2.1	68	68	177	A	1	XYZ	263	C	C	N	N	C	C	C	ABC	268	C	C	C	C	C	XYZ	*	C	C	C		
KM06'PROCESS'MAIL'MSG	JOV	XFT2.1	142	142	277	A	1	XYZ	263	C	C	N	N	C	C	C	ABC	268	C	C	C	C	C	XYZ	*	C	C	C		
KM07'INIT'GLOBAL'DATA	JOV	XFT2.1	85	85	113	A	1	XYZ	263	C	C	N	N	C	C	C	ABC	268	C	C	C	C	C	XYZ	*	C	C	C	C	
KM08'INIT'STARTUP'DATA	JOV	XFT2.1	45	45	92	A	1	XYZ	266	C	C	N	N	C	C	C	ABC	268	C	C	C	C	C	XYZ	*	C	C	C	C	
KM10'PROV'DZM'DATA	JOV	XFT2.1	42	42	101	A	1	XYZ	248	C	C	N	N	C	C	C	ABC	258	C	C	C	C	C	XYZ	***	C	C	C	C	
KM11'PROV'LDR'DATA	JOV	XFT2.1	69	69	165	A	1	XYZ	225	C	C	N	N	C	C	C	ABC	258	C	C	C	C	C	XYZ	***	C	C	C	C	
KM12'COMPUTE'LDR'DIST	JOV	XFT2.1	43	43	151	A	1	XYZ	225	C	C	N	N	C	C	C	ABC	258	C	C	C	C	C	XYZ	***	C	C	C	C	
KM13'COMPUTE'DZ'MARKER	JOV	XFT2.1	58	58	218	A	1	XYZ	225	C	C	N	N	C	C	C	ABC	258	C	C	C	C	C	XYZ	***	C	C	C	C	
KM14'COMPUTE'NE'REL'POS	JOV	XFT2.1	34	34	108	A	1	XYZ	225	C	C	N	N	C	C	C	ABC	258	C	C	C	C	C	XYZ	***	C	C	C	C	
KM15'COMPUTE'INTER'PARAMRS	JOV	XFT2.1	36	36	138	A	1	XYZ	225	C	C	N	N	C	C	C	ABC	258	C	C	C	C	C	XYZ	***	C	C	C	C	
KM40'PROCESS'MCD'INPUT	JOV	XFT2.1	88	88	195	A	1	XYZ	254	C	C	N	N	C	C	C	ABC	254	C	C	C	C	C	XYZ	***	C	C	C	C	
KM41'XMIT'SKE'DATA	JOV	XFT2.1	36	36	117	A	1	XYZ	254	C	C	N	N	C	C	C	ABC	254	C	C	C	C	C	XYZ	***	C	C	C	C	
KM42'INIT'SKE'BIT	JOV	XFT2.1	97	97	248	A	1	XYZ	254	C	C	N	N	C	C	C	ABC	254	C	C	C	C	C	XYZ	***	C	C	C	C	
KM43'SETUP'SKE'DATA	JOV	XFT2.1	48	48	190	A	1	XYZ	254	C	C	N	N	C	C	C	ABC	254	C	C	C	C	C	XYZ	***	C	C	C	C	
KM44'CONVERT'SLOT'NUMBER	JOV	XFT2.1	16	16	21	A	1	XYZ	254	C	C	N	N	C	C	C	ABC	254	C	C	C	C	C	XYZ	***	C	C	C	C	
VA00'GENERATE'VGD'COMMANDS	JOV	XFT2.0	95	95	256	A	1	AJR	212	C	C	C	N	C	C	C	PNQ	226	C	C	C	C	C	AJR	***	C	C			
VA00'GENERATE'VGD'COMMANDS	JOV	XFT2.1	25	112	291	A	1	AJR	265	C	C	C	N	C	C	C	PNQ	268	C	C	C	C	C	AJR	***					
VA03'COMPUTE'ELEVATOR'FEEDBACK	JOV	XFT2.1	100	72	236	A	1	AJR	266	C	C	C	N	C	C	C	PNQ	269	C	C	C	C	C	AJR	*					
VA04'INITIALIZE'VGD	JOV	XFT2.1	5	108	188	A	1	AJR	265	C	C	C	N	C	C	C	PNQ	268	C	C	C	C	C	AJR	***					
VA05'DETERMINE'ENG'DISENG'MODES	JOV	XFT2.1	10	60	200	C	1	AJR	***	C			N				PNQ	*						AJR	274					
VA08'INITIALIZE'EPR'PITCH	JOV	XFT2.1	5	33	74	A	1	AJR	265	C	C	C	N	C	C	C	PNQ	268	C	C	C	C	C	AJR	***					
VA10'GEN'FRONTSIDE'CMDS	JOV	XFT2.0	13	13	16	A	1	AJR	212	C	C	C	N	C	C	C	PNQ	224	C	C	C	C	C	AJR	***	C	C	C		
VA10'GEN'FRONTSIDE'CMDS	JOV	XFT2.1	5	18	23	A	1	AJR	266	C	C	C	N	C	C	C	PNQ	269	C	C	C	C	C	AJR	*					
VA20'GEN'BACKSIDE'CMDS	JOV	XFT2.1	15	59	252	A	1	AJR	265	C	C	C	N	C	C	C	PNQ	268	C	C	C	C	C	AJR	***					
VA30'GEN'SKE'SPEED'CMDS	JOV	XFT2.1	5	30	87	A	1	AJR	265	C	C	C	N	C	C	C	PNQ	268	C	C	C	C	C	AJR	***					
VA40'SET'STEERING'SUBMODE	JOV	XFT2.1	10	52	124	A	1	AJR	265	C	C	C	N	C	C	C	PNQ	268	C	C	C	C	C	AJR	***					
VA41'COMPUTE'SUBMODE'PARAMS	JOV	XFT2.1	5	38	161	A	1	AJR	265	C	C	C	N	C	C	C	PNQ	268	C	C	C	C	C	AJR	***					
VA47'SWITCH'FROM'DECK'ANGLE	JOV	XFT2.1	20	14	15	A	1	AJR	266	C	C	C	N	C	C	C	PNQ	269	C	C	C	C	C	AJR	*					
VA50'GEN'EPR'CMD	JOV	XFT2.1	5	28	91	A	1	AJR	266	C	C	C	N	C	C	C	PNQ	269	C	C	C	C	C	AJR	*					
VA60'GEN'PITCH'CMD	JOV	XFT2.1	5	43	181	A	1	AJR	266	C	C	C	N	C	C	C	PNQ	269	C	C	C	C	C	AJR	*					
VA70'GEN'NO'MODE'CMD	JOV	XFT2.1	5	20	32	A	1	AJR	266	C	C	C	N	C	C	C	PNQ	269	C	C	C	C	C	AJR	*					
VA90'GEN'AIRDROP'STAB'PITCH'CMD	JOV	XFT2.1	100	43	169	A	1	AJR	266	C	C	C	N	C	C	C	PNQ	269	C	C	C	C	C	AJR	*					
VB00'PERFORM'VERTICAL'GUIDANCE	JOV	XFT2.1	5	37	107	A	1	AJR	225	C	C	C	N	C	C	C	PNQ	267	C	C	C	C	C	AJR	***					
VB24'GEN'DRAG'MESSAGES	JOV	XFT2.1	5	108	197	A	1	AJR	268	C	C	C	N	C	C	C	PNQ	270	C	C	C	C	C	AJR	***					
VB50'OBTAIN'GLOBAL'INPUTS	JOV	XFT2.1	10	91	301	C	1	AJR	229	C	C	C	N	C	C	C	PNQ	***						AJR	***					

* = Due Week 273
*** = OVERDUE

Table 1. Weekly report of software units required for completion, with current status.

²All data shown in charts or tables is typical but not actual data.

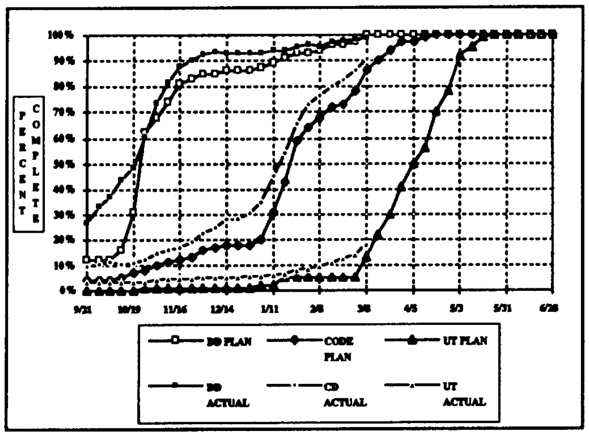
identifier, we plan, estimate, record and control its progress from the completion of top level design through the start of software integration. The unit name, build name (version), programming language (Jovial or Assembly), and programmer assigned to each implementation phase are entered into the database. Source Lines Of Code (SLOC) and memory size in 16-bit words are estimated at the outset of detailed design, to be updated after code compilation and linking. See Table 1[2].

The first key to successful project management is the ability to come up with a plan/schedule that is acceptable to both the customer and the software development staff. The bottoms up plan, estimated by software developers, is recorded in the data base as defined by the start and completion dates for each phase of each software unit planned to be developed. This plan is compared to the customer schedule requirements and the top down macro estimates based on overall software size and experienced software productivity rates. Negotiation, reallocation of resources and other tradeoffs are performed in an attempt to reach a consensus on a plan that can be supported by all stakeholders. Once this occurs the plan is baselined and progress against that plan is assessed.

Each programmer receives a weekly package of reports that allows him to plan his work to meet schedule requirements and to update the database with new status changes. Pages with the names of units whose detailed design, coding or testing phases are due for completion within the next two weeks, are distributed to those programmers who are responsible for those units, with flags indicating which phases are due for completion and which are overdue. Separate pages that indicate progress for each milestone completed in each unit are also distributed to the responsible programmers. By entering changes on these report pages, the database is updated weekly and new status reports are generated.

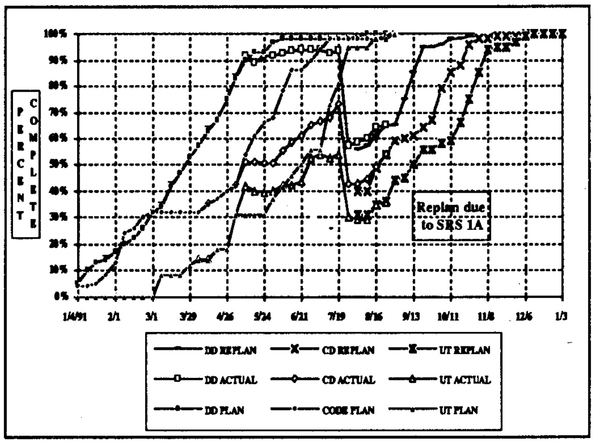
TYPICAL METRICS AND REPORTS PRODUCED

Our experience has demonstrated that the most effective metric from a project management viewpoint is the comparison of planned and actual progress curves for the three phases of implementation: detailed design, code and unit test. This graphic gives the best single picture of the software implementation status and a sense of how progress on early activities may affect later activities. It also allows a prediction of actual completion dates by extrapolation of the current slope of the actual progress lines. A typical example is shown in the chart below.



The plan lines on this chart are the cumulative SLOC-weighted planned completion dates for each unit as indicated by the scheduled completion date from the data base. Each unit completion is weighted by its size so as to give a more accurate estimation of the effort required. The plan lines are intended to represent the scheduled earned value of each phase. That is, scheduled or planned earned value is assumed to be proportional to estimated effort required. In general, the plan lines are only computed once or at least infrequently. The actual progress lines are updated every week based on the newly completed milestones for each unit in the data base, the weights of those milestones and the size of each referenced unit. It is important to note that each milestone is either complete or not—no partial credit is given. This is done to minimize subjectivity and ambiguity in the indicator. There must be a sufficient number of milestones tracked to provide the granularity required so that delays are not introduced by the policy of not allowing partial credit.

Once scheduled dates are frozen, only changes from original estimates in the size or number of units would cause the plan-line to change. Such changes would normally be trivial. However, when alterations in requirements cause a significant change in the work scope, a replan is generated and the schedule revised to fit the new requirements. We devised the concept of "effort SLOC" to quantify rework that involves rewriting code but does not result in additional delivered lines of code. Using effort SLOC to generate the data points that produce the plan-line, we are able to show breakage graphically, as shown in the following chart.



Using each programmer's weekly updated status of milestones completed, additional data points representing incremental work performed in each phase are generated and displayed graphically against the plan-line, providing instant feedback on performance versus schedule.

Summary reports indicating status of each software component, number of units, effort SLOC, deliverable SLOC, percentage of work complete in each phase, delta values between this week's and last week's status, historical summaries indicating progress over time, sizing summaries, listings of units placed in configuration management libraries, task-level summaries, and an assortment of other reports are produced weekly. Tables 2 and 3 represent typical reports.

MISSION COMPUTER SOFTWARE PROGRESS
SUMMARY OF MC OFP FT2.1 SOFTWARE UNIT DEVELOPMENT
WEEK 272 ENDING 20-SEP-91

CSC Name	Total SLOCS	Number of Units	% Detail Design Complete	% Code Complete	% Unit Test Complete
ADP					
BIT					
COM	220	2			
DTD	332	14	25.3		
EXE	20	2	5.0		
FMT	2351	57	6.1		
MDH	9848	159	85.8	66.3	14.9
MFD	1720	13	41.6		
MGF	496	4	100.0	100.0	100.0
NAV	1744	69	70.7	71.3	52.9
PSK	4934	54	100.0	100.0	84.4
RNZ	2118	34	100.0	100.0	100.0
VGD	635	24	66.9	51.1	
Total	24418	432	76.2	64.1	37.5

MISSION COMPUTER SOFTWARE PROGRESS
CHANGE SINCE WEEK 271 ENDING 13-SEP-91

CSC Name	Delta SLOCS	Delta Number of Units	% Delta Detail Design Complete	% Delta Code Complete	% Delta Unit Test Complete
ADP					
BIT					
COM					
DTD			25.3		
EXE					
FMT	1155	10	-5.9		
MDH	15	-1	1.8	10.5	4.0
MFD					
MGF					
NAV			0.2	2.0	
PSK					
RNZ					
VGD	290	5	-30.6	-43.1	
Total	1460	14	-3.2	0.7	-0.7

SLOCS = Delta Source Lines of Code

Table 2

WEEK 272 SUDS XFT2.1 SUMMARY BY TASK

CSC	TASK	Effort SLOCS in Task	Number of Units	Detail Design Percent Complete	Code Percent Complete	Unit Test Percent Complete
COM	CA00	100	1	0.0%	0.0%	0.0%
	CP00	120	1	0.0%	0.0%	0.0%
DTD	XA00	212	12	0.0%	0.0%	0.0%
	XB00	120	2	70.0%	0.0%	0.0%
EXE	ER00	10	1	5.0%	0.0%	0.0%
	ET00	10	1	5.0%	0.0%	0.0%
FMT	SD00	150	5	6.3%	0.0%	0.0%
	SM00	950	35	13.4%	0.0%	0.0%
	SS00			0.0%	0.0%	0.0%
	SZ00	96	7	6.3%	0.0%	0.0%
MDH	DK00	760	16	100.0%	67.2%	11.9%
	DKAL	40	2	100.0%	100.0%	0.0%
	DKEMT	470	3	100.0%	100.0%	25.0%
	DKFPL	1155	14	100.0%	90.0%	1.1%
	DKIOC	94	4	100.0%	100.0%	92.5%
	DKMNT	230	7	95.1%	2.1%	0.4%
	DKNAV	695	18	84.6%	29.3%	0.0%
	DKNDB	1020	40	100.0%	100.0%	0.0%
	DKPC	328	4	100.0%	100.0%	25.0%
	DKPCR	774	1	0.0%	0.0%	0.0%
	DKRNZ	1825	17	100.0%	74.9%	11.5%
	DKSKE	1020	11	100.0%	100.0%	70.0%
	DKSYS	350	7	100.0%	96.0%	20.1%
	DKTLD	532	10	46.0%	19.3%	15.6%
	DKVPF	555	5	61.9%	0.0%	0.0%
MFD	LA00	1520	12	34.0%	0.0%	0.0%
	LC00	200	1	100.0%	0.0%	0.0%
MGF	WD00	496	4	100.0%	100.0%	100.0%
NAV	JZ00	540	8	99.0%	96.2%	90.7%
	NA00	22	2	100.0%	100.0%	100.0%
	NC00	180	8	5.0%	0.0%	0.0%
	ND00	1002	51	66.7%	70.1%	41.0%
PSK	KA00	765	7	100.0%	100.0%	100.0%
	KB00	1051	13	100.0%	100.0%	100.0%
	KM00	3118	34	100.0%	100.0%	75.3%
RNZ	JZ00	13	1	100.0%	100.0%	100.0%
	RA00	374	6	100.0%	100.0%	100.0%
	RB00	1113	14	100.0%	100.0%	100.0%
	RC00	618	13	100.0%	100.0%	100.0%
VGD	VA00	310	14	100.0%	100.0%	0.0%
	VB00	320	9	34.3%	3.1%	0.0%
	VZ00	5	1	100.0%	100.0%	0.0%

Table 3

Tables 2 and 3. Typical software metrics reports produced weekly. Table 2 is an overall summary. Table 3 is a functional task-level report.

COST ESTIMATION FOR DEVELOPMENT PHASES

For budgetary and control purposes, software project activity elements are organized in a Work Breakdown Structure (WBS). Since the inception of the project, cost data for each element of software development process is being collected in accordance with the WBS. Archive tables of progress, and records of staffing for each phase of development, are being used to determine costs incurred for each element of the development effort as delineated by the WBS.

PROBLEMS OR DIFFICULTIES ENCOUNTERED

It is apparent that this approach requires thorough and detailed planning, probably more than most managers are accustomed to. It is our experience that the planning effort is well worth the investment and has payoffs in many different ways some of which may not have been expected.

Another consideration is the stability of the requirements for a particular software project. It is not inconceivable that changes may be incoming at such a high rate that the plan lines may change more than the actual progress which make project management nearly impossible. We have been able to accommodate a moderate amount of changes by grouping them into block changes, adjusting the plan lines once for the entire group and annotating the chart with the reason for the change. This approach has value in graphically demonstrating to the customer the price paid for lack of stability in the requirements.

One of the real keys to making this system work is the commitment and discipline of accurate weekly status updates to the data base. This is a long and difficult process for us. It was a significant cultural change to get the software development staff to come to realize that there was some greater value obtained by allowing the project to record not only their commitments in terms of the plan lines but also their actual performance against those commitments. A large percentage of the staff, particularly the "more experienced," had great difficulty in appreciating the value in this discipline. Eventually, after a number of years, this discipline was accepted by the majority of the staff. In fact, the public posting of these indicators, as shown in the picture below, led to a team commitment and pride in accomplishment when they saw their results meet the plan.

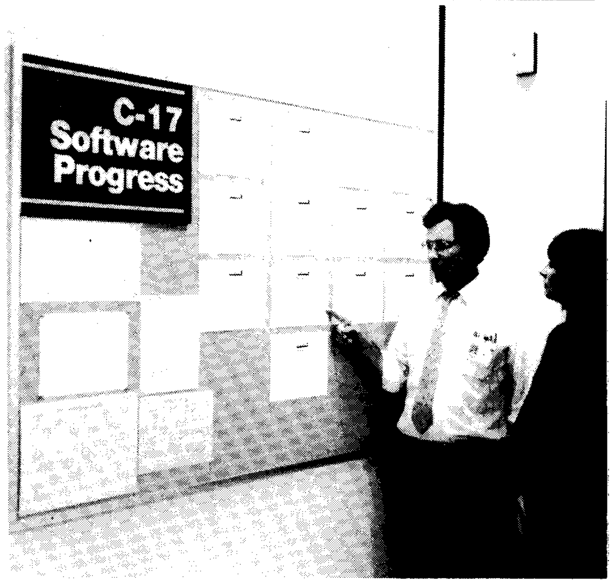
CONCLUSION AND LESSONS LEARNED

This detailed measurement system can be intimidating and ultimately unacceptable to the software development staff. Since this system does in fact document individual commitments there is danger of it being viewed as an invasion of privacy, and individual rating system or just another misdirected management project. By including everyone in the planning and by insuring that this data is never used to criticize individual performance, we ultimately gained the confidence and trust of the software development staff and their support of this measurement process.

It can be a serious mistake to publish a plan without comprehensive coordination and review, commitment from the software development staff and a good sense of any external impending factors. Any change in published plan lines reduces credibility in the system. Be very certain that your first release of a plan is realistic and that you have considered or accounted for any possible "surprises." There will be many reasons for changing the plan: data entry errors, second thoughts by developers, schedule slips of any supplier to the software process and, of course, changes in requirements. An important process is to accumulate and "block" changes in the requirements and then annotate the resultant change in the software plan lines. This make visible the price paid for changes and tends to reduce the level of this chronic problem for software managers. It can, however tend to increase any strain between the software development group and other parts of the organization responsible for software requirement definition.

It also creates a spirit of cooperation with the software quality assurance (SQA) group. Since many of the contractually required quality gates (reviews, walkthroughs and configuration management baselines) are part of the milestones tracked in the model, SQA has good visibility into the number and forecasted dates of activities they must audit. Use of the data base makes their planning and resource allocation easier to do. Conversely SQA's auditing of the progress data is an independent check on proper procedures and increases the confidence of software management.

Although public posting of plans and actual progress is a risk to a software team and its management, there are many benefits to this practice. Certainly, it can be an invitation to "micro-management" by other organizations and carries with it some risk of public failure, but on the other hand it demonstrates courage and shows everyone your success story when your team meets it commitments. We have posted not only the overall plans/actual lines but also individual software component subgroups plans/actual data. Of course this data is easy to obtain from our data base by a simple query. This provides a low level personal status reporting and introduces a healthy bit of inter-team competition. Public posting insures clarity of goals, fosters team commitment, and allows pride in accomplishment.



This system has provided an excellent response to numerous unplanned reviews, audits and investigative teams. It showed that we had a plan; it showed believable status with respect to that plan and we always received commendations for being well organized. Because of the regular weekly reporting and the flexibility of a relational data base we had the ability to produce a tailored, timely, high quality presentation to any group with less than 24 hours notice.

Finally and most fundamentally this system is an excellent software management tool. It provides great visibility into the processes, allows early detection of schedule problems enabling our managers to establish recovery plans in time to have a good probability of success and meet overall program requirements.