DRAFT 0.1

ACUS Metrics Program (AMP)

Ben Livson, 10 May 1989

Australian Centre for Unisys Software ACUS

Abstract

'When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind.'

Lord Kelvin, Popular Lectures and Addresses, 1889.

The goal of software engineering is accelerated production of higher quality software at lower cost. ACUS Metrics Program (AMP) has the goals of measuring this process and enabling ACUS management to monitor and to improve software engineering of ACUS products. Metrics are combined with hypotheses to produce models of software behaviour with the following practical benefits [1, p360]:

* The ability to describe quantitatively the current state of software parameters, such as software quality, resources expended, and productivity.

* The ability to predict software parameters, such as system cost, delivery time, and reliability.

* The ability to express requirements both quantitatively both as goals and as acceptance criteria.

* The ability to quantify trade-offs that can be used by management in allocating resources.

* The ability to monitor progress, to anticipate problems, and to provide feedback to software personnel about potential problems.

AMP covers metrics for

$ Cost, Schedule, Staffing, and Project Risk Estimation;

$ Reliability, Availability and Serviceability (RAS).

Cost, Schedule, Staffing, and Project Risk Estimation Metrics

ACUS has two different product lines:

1: Systems and communications software. Network Products (MHS, DSS, OSS, ISDN and SFO/ODA) and Solutions Generation Environment SGE User Administration and Directories components are projects characterized by tight, inflexible constraints and interface requirements. These projects are best described by the Constructive Cost Model (COCOMO) Embedded Development Mode [2].

2: Linc applications such as CareSys, and Oracle End User Interface OEUF projects are best modelled by Function Point Analysis [3,4,5]. A sanity check can be made using the COCOMO Basic Model Semidetached Development Mode.

The COCOMO model effort estimation model is: Effort = EAF*C*(KDSI)**EXP :

The Effort Adjustment Factor EAD is the product of the Effort Multipliers corresponding to each of the fifteen cost drivers (Analyst Capability, Analyst Experience, Product Complexity, Database Size, Language Experience, Modern Programming Practices, Programmer Capability, Required Software Reliability, Required Development Schedule, Main Storage Constraint, Execution Time Constraint, Usage of Software Tools, Computer Turnaround time, and Virtual Machine Volatility). The Basic Model does not use cost drivers (EAD=1).

Constant C is set according to the development mode (Organic, Semidetached or Embedded) and according to the selected COCOMO model (Basic Model, Intermediate and Detailed Model). The basic input is thousands of delivered source instructions KDSI. Exponent EXP depends on the development mode used.

The COCOMO detailed model uses different Effort Multipliers for the six life-cycle phases (differ from Unisys phases):

Phase-1: Requirements;
Phase-2: Product Design
Phase-3: Detailed Design
Phase-4: Code and Unit Test
Phase-5: Integrate and Test
Phase-6: Maintenance

ACUS Network Products Group has used the COCOMO model manually. Software Systems Costar tool [6] was selected. Costar is a full implementation of COCOMO. Costar also produces special estimates for adapted and ported software. Costar 3.00 includes Function Point Analysis as well. Costar produces the following reports (tables and plots):

$ Development Summary
$ Cost Profile Graph (the estimated cost in each month)
$ Staffing Profile Graph
$ Schedule & Detail Reports
$ Activity Report helps match staff to activities in each phase
$ Comparison Report (several estimations are displayed in parallel)

Estimation breakdown to software components is enabled. Costar is used for successive project estimation refinements. The tool can be calibrated for AMP.

Function Point Analysis FPA [3] defines productivity in terms of a weighted sum of delivered functional units. Functional units are defined as the number of inputs, the number of outputs, the number of inquiries, and the number of files. The method was originally developed by A. Albrecht for IBM's Data Processing Division and applied best for DP/LOB applications. Function points for systems and network software are difficult to define precisely and, in any case may differ significantly in size and scope. ACUS has used the Estimacs [4] tool for Function Point Analysis. Estimacs produces reports similar to Costar to estimate cost, schedule, staffing and risk. The Estimacs model is not publicly documented but the tool lists the most significant cost drivers and enables sophisticated sensitivity and risk "What -- IF" analysis. Estimacs has parameters covering organizational influence.

Unisys PCFPA [5] helps counting function points per each screen and report of a DP/LOB application. Typically, project estimation starts with Estimacs and continues using PCFPA.

Both Costar and Estimacs data can be easily entered to a Project Planning Tool, say PMW [7], used for staffing and scheduling project activities. In fact, Costar and Estimacs data can be output to flat files which in turn can be processed by utilities to feed PMW for whatever this is worth. Reversing this process is more difficult.

Project estimation tools are known to produce very large errors. The quality of estimates, of course, depends on the accuracy of input parameters. Two different estimation models, say COCOMO and FPA, should always be used as a sanity check. Never trust one model. Produce a set of estimates for sensitivity and risk analysis. Store your estimates and compare current estimates with the previous iterations. Calibrate your tools to match your project history.

Responsibility: Project Management has the primary responsibility for using project estimation tools and generating the associated metrics.

Reliability, Availability and Serviceability (RAS) Estimation Metrics

Unisys RAS program [8,9,10] implementation for the ACUS Metrics Program is presented. Support phase Service System Objectives [10] are compulsory. Interpretation and implementation of the RAS program itself is a voluntary process and each development centre should be able to decide what data will be collected and what metrics will be calculated. ACUS approach is to use the Engineering PRIMUS Application System EPAS [15] as a qualification and support phases 4-5 database. Creation of Product Validation Profiles is pushed forwards to phases 2-3 enabling collection of development phase RAS data. However, use of EPAS prior to qualification is not compulsory. All serviceability metrics and most reliability & availability metrics will be extracted from EPAS. Corporate RAS program does not cover critical metrics such as Readiness for Qualification and Readiness for Release (FCS). These and a number of other metrics have been modelled for AMP using the IEEE Standard Measures to Produce Reliable Software [11].

Basic Corporate RAS Metrics Set

M1: Reported Arrival Quality

RAS Category: Installability

Recommended Metrics: Style-ID Arrival Quality = (1 or 0) where

0 if any item in the Bill of Materials for this Style-ID is either not received, damaged, or fails to install;

1 otherwise.

ORDER ARRIVAL QUALITY = SUM (Style-ID Arrival Quality) / #Style-IDs.

MEAN REPORTED ORDER ARRIVAL QUALITY PER PERIOD = SUM (ORDER ARRIVAL QUALITY) / #PERIOD-ORDERS

Read # as "Number of".

Discussion: Arrival Quality is the customer's first impression of Unisys and as such critical. Collection of the metrics data is complicated. Corporate (EAD/PAD for CareSys) marketing has supply data on orders per period. Data on arrival quality problems should be reported as PRIMUS UCFs. ACUS does not have any other established mechanism. Problems related to CSPO manufacturing and deliveries may never be raised as UCFs in which case ACUS values for reported order quality may be too high.

Resposibility: ACUS Support/CM.

M2: Rate of Reported Problems/NFS

RAS Category: Reliability

Recommended Metrics:

UCFs per PERIOD
UCFs per PERIOD per CURRENT USER
PRIORITY-X UCFs per PERIOD, X=A,B,C
PRIORITY-X UCFs per PERIOD per CURRENT USER
PRIMUS CONTACTS per PERIOD
PRIMUS CONTACTS per PERIOD per CURRENT USER

where CURRENT USERS = #Shipments - #Upgrades.

Discussion: All data is available from EPAS. The metrics should be calculated both monthly and between releases. Compute the metrics separately for problems and new feature suggestions. Extract all UCFs raised by ACUS.

Responsibility: ACUS Support/CM.

M3: Current Quality Indicator

RAS Category: Reliability

Recommended Metrics:

The tuple (#A, #B, #C) Unique Open Problems

where a Unique Problem is a PRIMUS Problem List Entry PLE raised against customer UCF. Extract UCFs raised by ACUS against ACUS. The metric is extracted by cross selection of open UCFs against PLEs with UCF type Released, Priority=A,B,C and FORM=Trouble Report. Compute Quality Indicator per release and monthly. Compute this metrics per UCF class (Software or Documentation) and per major product components. Compute separate metrics for ACUS Development UCFs and Internal PLEs. Note that a PLE need not be raised against a type Development UCF.

Responsibility: ACUS Support/CM.

M4: Incident Encounter Rate

This metrics cannot be used as part of AMP, because ACUS does not have access to data on customer machine/product operating hours. This metrics may be used internally as part of ACUS product qualification.

M5: Machine Stop Rate per Product (not applicable for AMP. See M4 explanation)

M6: Software Service Cost

RAS Category: Maintainability

Recommended Metrics: Service Cost Per Fix Per Period =

Cost of Product Support / # Fixes per Period

Cost Per Shipment Per Period = SUM of Support Cost per Period / #Shipments

Similarly, calculate lifetime service cost per shipment per feature release.

Responsibility: ACUS Support.

M7: UCF Service Response Time

RAS Category: Serviceability

Recommended Metrics: Priority (A, B, C) UCF RESPONSE TIME

Priority (A, B, C) UCF CLOSURE TIME

Priority (A, B, C) UCF CORRECTION AVAILABILITY TIME

Discussion: Correction Availability is defined as the elapsed tome from when the UCF is received by Unisys until when the problem correction is delivered to customer and customer satisfaction is verified. Unisys Service System Objectives mandate a UCF engineering response within 15 days to priority A UCFs and within 45 days to priority B UCFs. UCFs must be closed within 180 days. In fact, priority A UCFs should have very short cycles and AMP shall constantly monitor UCF Service Response Time. Observe that a more detailed analysis of UCF closure time shall be carried out against UCF Closure Codes.

Responsibility: ACUS Support.

In summary, AMP implements the basic corporate RAS metrics with the exception of metrics M4 and M5 that can be applied only for in-house prerelease estimation.

Recommended Additional RAS Metrics

AMP collects installation time sample statistics with the help of customer support centres and subsidiaries, and statistics about complaints per shipment. ACUS support is responsible. Statistics of mean calendar hours, staff hours and problems per installation will be collected by ACUS support.

Also, PRIMUS PLE statistics recommended as additional RAS metrics will be part of AMP. RAS hardware metrics [8, appendix B] and availability and downtime metrics [8, appendix C] are not applicable for AMP.

Unique AMP RAS Metrics

The unique AMP RAS metrics are RAS metrics for pre FCS metrics not covered by the corporate program. The IEEE standard [11] for Producing Reliable Software provides a detailed discussion of these metrics. Only the metrics applicable to AMP are listed. Only a small subset of [11] is used. In particular, the stochastic metrics for predicting reliability growth and remaining problems by analysing problem incident times will not be used during the initial AMP. The reasons for not including stochastic metrics are that data collection usually starts only at the beginning of qualification and qualification periods are usually very short which render the stochastic measures meaningless. On the other hand, AMP expands on RAS metrics using error seeding to derive unique metrics for unit, integration, system, acceptance test and release readiness.

U1: Fault Density

Metrics: #PLE / KDSI and #UCF / KDSI.

where KDSI is thousands of delivered source instructions. Calculate this measure per PLE and UCF type and priority, status and closure code. Measure U1 extends measure M3.

Responsibility: Product Assurance and Support.

U2: **Inspection Problem Density**

Number of problems per Number of Specification Pages or per Source Lines as applicable for inspections, audits, reviews and walk-throughs. Collect statistics to show product trends. Also, log manhours spent on inspections to measure manhours per problem detected.

Responsibility: Product Assurance

U3: **Functional Test Coverage**

Functional Test Coverage = #Functions Tested / #Functions

Responsibility: Development and Product Assurance

U4: **Requirements Traceability**

requirements met by product during acceptance test / # original requirements

Responsibility: Product Assurance

U5: **Software Maturity**

Software Maturity = (#function - #function(added+changed+deleted))/#function

Responsibility: Product Assurance

U6: **Run Reliability**

Run reliability R(k) is the probability that k randomly selected runs will produce a correct result. The problem applying this measure is how to randomly invoke runs over the entire input space of all possible input patterns and states. A run is a test case. If an invocation probability for each test case can be assigned and if the test cases represent the equivalence classes of the input space, then run reliability is easily computed.

Responsibility: Development Product Assurance

U7: **Test and Release Readiness Metrics - By Problem Seeding**

Primitives:

J = the number of problems intentionally seeded (inserted)
j = the number of seeded problems found
n = the number of faults found that were not intentionally seeded

The maximum likelihood estimate of the number of indigenous (unseeded) problems remaining in the product is: $N = nJ/j$.

Discussion: The number of seeded and indigenous problems enables the number of problems remaining to be estimated for the problem type being considered. In particular, the problems seeded should include program unit test problems, program unit interface problems to check integration testing, installation problems, functional problems of missing requirements, misinterpreted requirements and functions not documented by specifications, system test performance problems, error handling problems, user interface and documentation problems. The problems seeded must be representative and non-trivial. Before seeding, a fault analysis is needed to determine the types of errors and their relative frequency of occurrences. The accuracy of the estimate (and hence the confidence in it) increases as the number of seeded problems increases.

Faults should be inserted randomly throughout the software. Personnel inserting the faults should be different and independent of those persons later searching for the faults. The process of searching should be carried out without knowledge of the inserted problems.

Good configuration management is to assure that the seeded problems are not left in the final version released and that indigenous problems are corrected only in the unseeded version.

Error seeding can be used to prove exit readiness from unit, integration & system test, functional, installation, acceptance and field tests. In particular, exit from a test regime should not be allowed should there remain priority A seeded problems undetected. Such a situation clearly indicates that testing should continue or that testing personnel are incapable of doing their job successfully. Project testing and qualification risks could be reduced by making limited trials with seeded problems before commencing any major testing activity. In this way we can be confident in our testing personnel.

Based on the estimated number of indigenous problems per thousands of source instructions delivered and per problem priority we can decide whether a release should be approved given that acceptance test criteria has been established as part of the acceptance test plans.

Problem seeding seems to be particularly effective for evaluating the efficiency of third party beta-sites. There has to be, of course, agreement with Program Management and beta-sites that error seeding is permitted.

Responsibility: Development and Product Assurance

U8: Quality Assessment

Product Assurance is responsible for defining criteria for project quality assessment and evaluation this criteria at least as part of the phase review process. Each criteria can be given a weight and scale. The process can be supported by a tools such as POWER.

Project Observation Workbench and Evaluation Reporter POWER [12] automates and standardizes the collection and evaluation of software project status information. POWER uses a series of over 600 structured questions to evaluate the state of a project during the six Phases of project development. The questions have been weighted by POWER according to the size of the software under development, its complexity, and whether the program is real time or non-real time. You can modify the POWER question set to meet your particular needs. POWER has been used to evaluate the ACUS ODA/SFO project. Its use is regarded as tedious.

U9: Complexity and Software Science Metrics

Product Assurance is responsible for introducing tools that analyse source code and (semi)formal specifications for complexity metrics that can be automatically extracted. Complexity metrics can be used to find overly complex software for redesign, to determine problematic modules and to allocate testing and maintenance resources.

SOFMET [13] is a software metrics tool that produces Halstead, Lines of Code, McCabe and implementation weight/corrected volume metrics for OS1100 design and programming languages (Structured Design Language SDL, SSG, PLUS, FORTRAN, MASM/ASM, COBOL, C and INLINE).

A Unix and PC DOS based third party tool Qualigraph [14] provides similar metrics for most popular programming languages.

Halstead's Software Science Measures [11] primitives are:

n1 = # distinct operators
n2 = # distinct operands
N1 = # operators
N2 = # operands

The metrics are:

Program Vocabulary: n = n1 + n2
Observed Program Length: N = N1 + N2
Estimated Program Length: N*= n1(log n1) + n2 (log n2) ; log base =2
Program Volume: V = N (log n)
Program Difficulty: D = (n1 / 2)(N2 / n2)
Program Level: L = 1/D
Effort: E = V/L
#Problems: B = V/3000 =~ E**0.67/3000
Time: T = E/S (S = Stroud number, typically 18 elementary mental discriminations per second).

Cyclomatic McGabe Complexity primitives [11] are:

n = #nodes (sequential groups of program statements)
e = #edges (program flows between nodes)
s = #splitting nodes (node with more than one edge emanating from it).
r = #regions (areas bounded by edges with no edges crossing)

A strongly connected graph is required. A strongly connected graph is one in which a node is reachable from any other node: this is achieved by adding an edge between the exit node and the entry node.

The cyclomatic complexity is $r = e - n + 1 = s + 1$.

Values greater or equal to ten per program unit are regarded problematic. A strong correlation exists between cyclomatic complexity and software problems.

Corrected Volume is the product of a module's Halstead Volume and its Cyclomatic Complexity. Implementation Weight is the sum for all modules scanned.

REFERENCES

Unisys restricted references are marked [*].

[1] Software Engineering Metrics and Models, Conte-Dunmore-Shen, The Benjamin/Cummings Publishing Company, Inc. 1986.

[2] Software Engineering Economics, Barry Boehm, Prentice-Hall, 1981.

[3*] Burroughs Function Point Analysis Guide, EP9902.

[4] CA-Estimacs User Guide, Computer Associates International, Inc.

[5*] Unisys PCFPA Project Estimation Software Version 2.1, Creg Hendrics, Customer Service, Industrial and Commercial Division, 122 West Carpenter Freeway, Irrving, TX 75039, (214) 659-8134.

[6] Costar User Documentation, Softar Systems, 28 Ponemah Road, Amherst, NH 03031, (603) 672-0987.

[7] Project Manager Workbench, User Documentation, PMW Centre, Hoskyns Group plc, 130 Shaftesbury Avenue, London W1V 7DN, 01-434 2171.

[8*] Corporate Software RAS Metrics, Draft Version B, May 1988.

[9*] Corporate Industry Management PA & S Manager's Task Force Metrics Project, September 7, 1988. Contact: Jennysue Mott, Charlotte CDC.

[10*] UCF Policy & Service Objectives, Pam Murray, Blue Bell SDG, 1989.

[11] ANSI/IEEE Standard 982.1-1988 and Guide 982.2-1988 Measures to Produce Reliable Software.

[12] POWER (tm) System Manager's Manual, Expertware, Inc., 2685 Marine Way, Suite 1209, Mountain View, CA 94043, (415)965-8921.

[13*] SOFMET 1100 Level 1R7 User Guide, RSS-160, Clear Lake 1989 March 10.

[14] Qualigraph(tm) User Documentation, Micro Data Management Systems, SZKI-Computer Research and Innovation Centre.

[15*] Engineering PRIMUS Application System (EPAS), User Reference, UCF Coordinator Roseville, 1989 February 02.