1.1. Cost Estimation Models

Parametric cost models used in avionics, space, ground, and shipboard platforms by the services are generally based on the common effort formula shown in Equation 1. Size of the software is provided in a number of available units, cost factors describe the overall environment and calibrations may take the form of coefficients adjusted for actual data or other types of factors that account for domain-specific attributes [Lum et al. 2001], [Madachy and Boehm, 2008]. The total effort is calculated and then decomposed by phases or activities according to different schemes in the models.

$$Effort = A * Size^{B} * EM \; . \qquad (1)$$

- Effort in person-months
- A - calibrated constant
- B - scale factor
- EM - effort multiplier from cost factors

The popular parametric cost models in widespread use today allow size to be expressed as lines of code, function points, object-oriented metrics and other measures. Each model has its own respective cost factors for the linear effort multiplier term, and each model specifies the B scale factor in slightly different ways (either directly or through other factors). Some models use project type or application domain to improve estimating accuracy. Others use alternative mathematical formulas to compute their estimates. Sizing for the models are covered in Chapter 2. A comparative analysis of cost models is provided in Section TBS. The model WBS phases and activities are addressed in Section TBD.

The models covered include COCOMO II, SEER-SEM, SLIM, and True S. They were selected because they are the most frequently used models for estimating DoD software effort, cost and schedule. A comparison of the COCOMO II, SEER-SEM and True S models for NASA projects is described in [Madachy-Boehm 2008]. A previous study at JPL analyzed the same three models with respect to some of their flight and ground projects [Lum et al. 2001].

The consensus of these studies is any of the models can be used effectively if it is calibrated properly. Each of the models has strengths and each has weaknesses. For this reason, the studies recommend using at least two models to estimate costs whenever it is possible to provide added assurance that you are within an acceptable range of variation.

Other industry cost models such as SLIM, Checkpoint and Estimacs have not been as frequently used for defense applications as they are more oriented towards business applications per [Madachy-Boehm 2008]. A previous comparative survey of software cost models can also be found in [Boehm et al. 2000b].

COCOMO II is a public domain model that USC continually updates and is implemented in several commercial tools. True S and SEER-SEM are both proprietary commercial tools with unique features but also share some aspects with COCOMO. All three have been extensively used and tailored for flight project domains. SLIM is another parametric tool that uses a different approach to effort and schedule estimation.

1.1.1. COCOMO II

The COCOMO (COnstructive COst MOdel) cost and schedule estimation model was originally published in [Boehm 1981]. The COCOMO II research effort was started in 1994, and the model continues to be updated at USC, the home institution of research for the COCOMO model family. COCOMO II defined in [Boehm et al. 2000] has three submodels: Applications Composition, Early Design and Post-Architecture. They can be combined in various ways to deal with different software environments. The Application Composition model is used to estimate effort and schedule on projects typically done as rapid application development. The Early Design model involves the exploration of alternative system architectures and concepts of operation. Typically, not enough is known to make a detailed fine-grained estimate. This model is based on function points (or lines of code when available) and a set of five scale factors and seven effort multipliers.

The Post-Architecture model is used when top level design is complete and detailed information about the project is available and the software architecture is well defined. It uses Source Lines of Code and/or Function Points for the sizing parameter, adjusted for reuse and breakage; a set of 17 effort multipliers and a set of five scale factors that determine the economies/diseconomies of scale of the software under development.

The effort formula is:

$$Effort = A * Size^{B} * \prod_{i=1}^{N} EM_i \,. \qquad (2)$$

Where
- *Effort* is in person-months
- *A* is a constant derived from historical project data
- *Size* is in KSLOC (thousand source lines of code), or converted from other size measures
- *B* is an exponent for the diseconomy of scale dependent on additive scale drivers
- EM_i is an effort multiplier for the i^{th} cost driver. The geometric product of *N* multipliers is an overall effort adjustment factor to the nominal effort.

The COCOMO II effort is decomposed by lifecycle phase and activity as detailed in following model comparisons.

1.1.2. True S

True S is the updated product to the PRICE S model offered by PRICE Systems. PRICE S was originally developed at RCA for use internally on software projects such as the Apollo moon program, and was then released in 1977 as a proprietary model. It fits into a composite modeling system and can be used to estimate more than just software costs. Many of the model's central algorithms were published in [Park 1988]. For more details on the model and the modeling system see the PRICE Systems website at http://www.pricesystems.com.

The PRICE S model consists of three submodels that enable estimating costs and schedules for the development and support of computer systems. The model covers business systems, communications, command and control, avionics, and space systems. PRICE S includes features for reengineering, code generation, spiral development, rapid development, rapid prototyping, object-oriented development, and software productivity measurement. Size inputs include SLOC, function points and/or Predictive Object Points (POPs). The True S system also provides a COCOMO II capability.

The switch to True S is currently taking place. Hence some of the descriptions retain the old PRICE S terminology (such as the Rosetta Stone) while we move towards a complete True S implementation. Numeric estimate results shown are for the latest True S model.

The TruePlanning estimation suite from PRICE Systems contains both the True S model and the COCOMO II cost model.

1.1.3. SEER-SEM

SEER-SEM is a product offered by Galorath, Inc. This model is based on the original Jensen model [Jensen 1983], and has been on the market over 15 years. The Jensen model derives from COCOMO and other models in its mathematical formulation. However, its parametric modeling equations are proprietary. Like True S, SEER-SEM estimates can be used as part of a composite modeling system for hardware/software systems. Descriptive material about the model can be found in [Galorath-Evans 2006].

The scope of the model covers all phases of the project life-cycle, from early specification through design, development, delivery and maintenance. It handles a variety of environmental and application configurations, and models different development methods and languages. Development modes covered include object oriented, reuse, COTS, spiral, waterfall, prototype and incremental development. Languages covered are 3rd and 4th generation languages (C++, FORTRAN, COBOL, Ada, *etc.*), as well as application generators.

The SEER-SEM cost model allows probability levels of estimates, constraints on staffing, effort or schedule, and it builds estimates upon a knowledge base of existing projects. Estimate outputs include effort, cost, schedule, staffing, and defects. Sensitivity analysis is also provided as is a risk analysis capability. Many sizing methods are available including lines of code and function points. For more information, see the Galorath Inc. website at http://www.galorith.com.

1.1.4. SLIM

The SLIM model developed by Putnam is based on a Norden/Rayleigh manpower distribution and his finding in analyzing many completed projects [Putnam and Myers 1992]. The central part of Putnam's model called the *software equation* is:

$$S = C_k * Effort^{1/3} * t_d^{4/3} . \qquad (3)$$

Where

- *Effort* is in person-months
- t_d is the software delivery time;
- C_k is a productivity environment factor.

The productivity environment factor reflects the development capability derived from historical data using the software equation. The size S is in LOC and the *Effort* is in person-years. Another relation found by Putnam is

$$Effort = D_0 * t_d^3 . \qquad (4)$$

where D_0 is the *manpower build-up parameter* which ranges from 8 (entirely new software with many interfaces) to 27 (rebuilt software). Combining the above equation with the software equation, we obtain the power function forms for effort and schedule:

$$Effort = (D_0^{4/7} * E^{-9/7}) * S^{9/7} . \qquad (5)$$

$$t_d = (D_0^{-1/7} * E^{-3/7}) * S^{3/7} . \qquad (6)$$

Putnam's model is used in the SLIM software tool based on this model for cost estimation and manpower scheduling [QSM 2003].

1.2. Lifecycles and Metrics Collection

Discuss baseline lifecycle models, the milestones, endpoints, activities, WBS implications

Traditional Waterfall Model

Rational Unified Process Model

Rational Unified Process (RUP) phases of Inception, Elaboration, Construction, and Transition (IECT):

Comparison of Waterfall to RUP

Challenges in collecting data, e.g. CM effort may be buried in other activities

Different life cycle paradigms can complicate matters because influence timing and allocation of effort to activities. For example, activities may be repeated and their products may be refactored should they be generated incrementally in parallel or in spirals. Under such cases, code size has to be adjusted to include code used from previous versions that incorporates no changes what-so-ever. This “used” code is integrated and used as-is instead of being instantiated for reuse.

TBD (Issues: overlapping development effort, overlapping phases, inclusion of non-charged activities,e.g. CM, QA, PI)

Challenges in mapping cost estimation approaches to software development lifecycles

TBD (Issues: models have a defined breadth and depth, breaking up model estimates,

If agile methods like extreme programming [2] or Scrum [3] be employed, traditional requirements specifications may not be generated and test-first concepts may be used to ensure the product meets warfighter needs. For these developments, the software is managed as builds, not components, and is generated throughout the project as a series of increments to address risk via a series of production prototypes.

Testing occurs at the build and not the component level. This complicates the Work Breakdown Structure because tasks have to be expressed in terms of the life cycle utilized.

overlapping model estimates)

How do we account for layers of integrations, e.g. system of systems.

1.3. Cost Model Lifecycles and Work Breakdown Structures

COCOMO II allows effort and schedule to be allocated to either a waterfall or MBASE lifecycle. MBASE is a modern iterative and incremental lifecycle model like the Rational Unified Process (RUP) or the Incremental Commitment Model (ICM). The phases include: (1) Inception, (2) Elaboration, (3) Construction, and (4) Transition.

True-S uses the nine DoD-STD-2167A development phases: (1) Concept, (2) System Requirements, (3) Software Requirements, (4) Preliminary Design, (5) Detailed Design, (6) Code/Unit test, (7) Integration & Test, (8) Hardware/Software Integration, and (9) Field Test.

In SEER-SEM the standard lifecycle activities include: (1) System Concept, (2) System Requirements Design, (3) Software Requirements Analysis, (4) Preliminary Design, (5) Detailed Design, (6) Code and Unit Test, (7) Component Integration and Testing, (8) Program Test, (9) Systems Integration through OT&E & installation, and (10) Operation Support. Activities may be defined differently across development organizations and mapped to SEER-SEMs designations.

In SLIM the lifecycle maps to four general phases of software development. The default phases are: 1) Concept Definition, 2) Requirements and Design, 3) Construct and Test, and 4) Perfective Maintenance. The phase names, activity descriptions and deliverables can be changed in SLIM.

The phases covered in the models are summarized in Table 1, activities in Table 2 and their categories of cost sources in Table 3.

Table 1: Model Lifecycle Phases

Model	Phases
COCOMO II	• Inception • Elaboration • Construction • Transition • Maintenance
SEER-SEM	• System Requirements Design • Software Requirements Analysis • Preliminary Design • Detailed Design • Code / Unit Test • Component Integrate and Test • Program Test • System Integration Through OT&E
True S	• Concept • System Requirements • Software Requirements • Preliminary Design • Detailed Design • Code / Unit Test • Integration and Test • Hardware / Software Integration • Field Test • System Integration and Test • Maintenance
SLIM	• Concept Definition • Requirements and Design • Construction and Testing • Maintenance

Table 2: Model Cost Activities

Model	Activities
COCOMO II	• Management • Environment / CM • Requirements • Design • Implementation • Assessment • Deployment
SEER-SEM	• Management • Software Requirements • Design • Code • Data Programming • Test • CM • QA
True S	• Design • Programming • Data • SEPGM • QA • CFM
SLIM	• WBS Sub-elements of Phases: ○ Concept Definition ○ Requirements and Design ○ Construct and Test ○ Perfective Maintenance

Table 3: Model Cost Categories

Model	Categories
COCOMO II	• Software Engineering Labor
SEER-SEM	• Software Engineering Labor • Purchases
True S	• Software Engineering Labor • Purchased Good • Purchased Service • Other Cost
SLIM	• Software Engineering Labor

The vendor tools also provide utilities to import or define alternative work breakdown structures and activities that are decomposed by effort and schedule percentages. The SEER-SEM and True S tool suites cover other discipline such as systems and hardware engineering, and COCOMO II has a suite of other models covering systems engineering and more.