Collecting Data for the New COCOMO® III

Brad Clark, PhD

20th Practical Software and Systems Measurement Users' Group Workshops

September 18, 2019

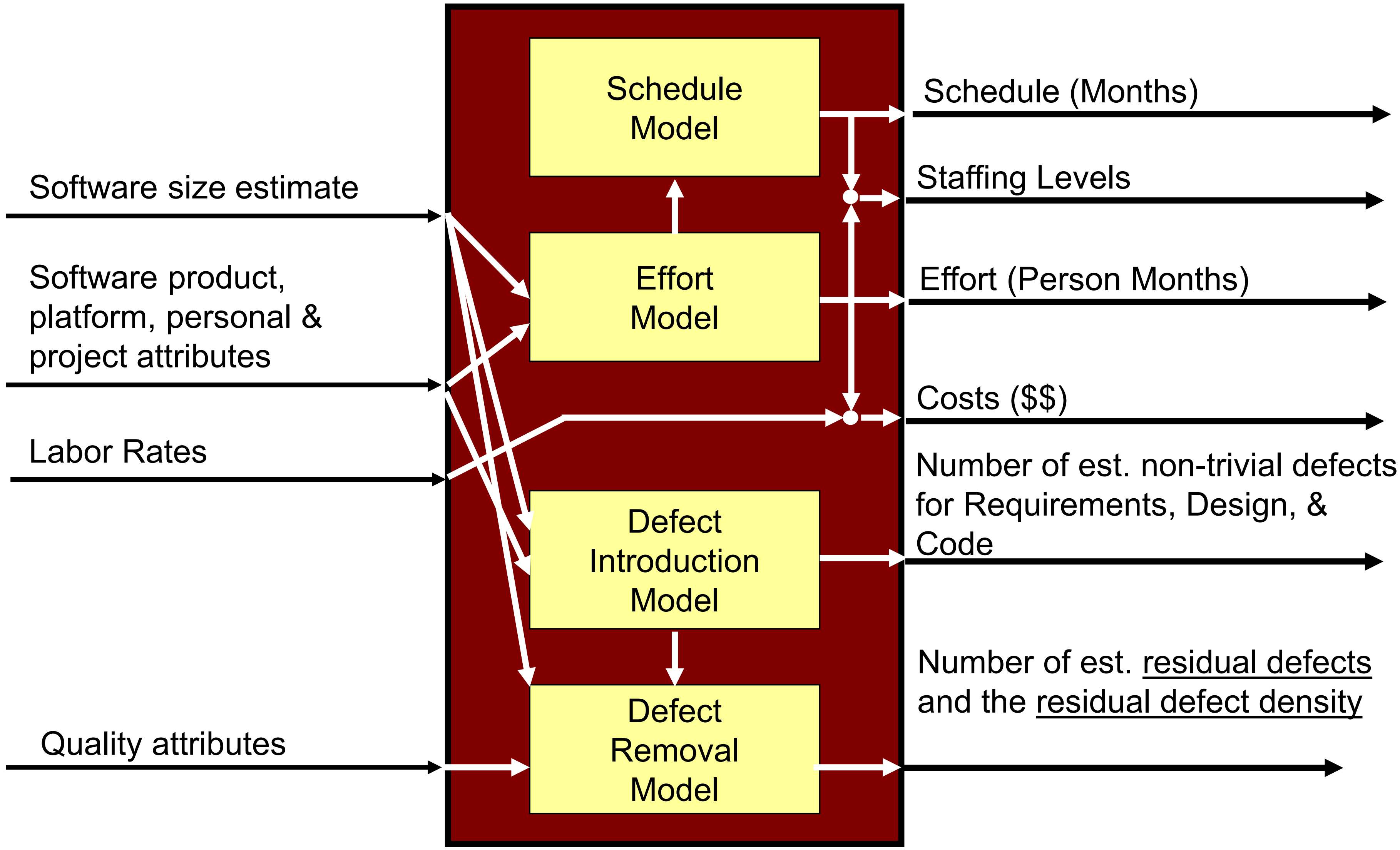
COCOMO® III

- The COCOMO III model is an update on the popular COCOMO II Software Cost Estimation Model.
- A draft version of the model has been formulated and the next step is to calibrate the model to real-world data.
- The updates to the new model include
  - functional size inputs
  - a new Software Security parameter
  - removal of a couple of COCOMO II parameters
  - an update to some of the pre-existing COCOMO II parameters.

The COCOMO® III Project

- It has been 19 years since the COCOMO II model has been updated and calibrated to new Software Engineering practices
- COCOMO family of models has grown:
  - COSYSMO
    - Interact with the System Engineering Cost Estimate
  - COQUALMO
    - Quality model incorporation into COCOMO III
- Introduction of model "Pre-sets"
- Expand size measure inputs
- "Nominal" rating has shifted for some drivers
- Some drivers are new, split, changed, or dropped

COCOMO III Model Concept


Software size estimate
Software product, platform, personal & project attributes
Labor Rates
Quality attributes
Schedule Model
Effort Model
Defect Introduction Model
Defect Removal Model
Schedule (Months)
Staffing Levels
Effort (Person Months)
Costs ($$)
Number of est. non-trivial defects for Requirements, Design, & Code
Number of est. residual defects and the residual defect density

Workload Sizing

- The amount of development work to be done is expressed as either a functional or product size
  - Software Requirements
  - Function Points
  - SNAP Points
  - Fast Function Points
  - COSMIC Points
  - Automated Function Points
  - Feature Points
  - Use Case Points
  - Story Points (*Agile Development*)
  - Source Lines of Code
- The desire is for COCOMO III to use different size types organically as a size input
  - Want to move away from converting one size type to another, e.g. Function Points to Source Lines of Code

Data Collection

- The COCOMO III model development is sponsored by the University of Southern California Center (USC) for Systems and Software Engineering (CSSE).
- A spreadsheet is used to collect data
- There are three data collection tabs:
  - System Context
  - Component Info
  - Component Size
- There are two definition tabs:
  - Cost Drivers
  - Application Super-Domains

Data Management & Protection

- Data is submitted in a <u>sanitized format</u>. The data provider will choose sanitized system and component names only known to them
- After data submission, a telecon will be schedule to clarify any issues
- All data is encrypted and stored in a secure cloud repository protected by strong passwords
- Data access is restricted to the CSSE COCOMO III Data Manager/Analyst
- Sanitized, abridged data will be provided on a limited basis to CSSE Researchers
- Data will only be displayed publicly in a summarized format using all of the data, e.g., scatter plots, histograms

System Context

- System Name (Sanitized)
- System Description / Architecture
- Application Super-Domain (RT, ENG, AIS)
- # Unique Software Baselines
- Development Process
- Development Iteration
- System Comments
- Data Submitter Point of Contact

System-level Cost Drivers

- Precedentedness
- Development Flexibility
- Risk/Opportunity Management
- Software Architecture Understanding
- Stakeholder Team Cohesion
- Process Capability & Usage
- Required Development Schedule

System Driver Example

Software Architecture Understanding (ARCH)

This driver rates the degree of understanding of determining and managing the system architecture in terms of platforms, standards, new and NDI (COTS/GOTS) components, connectors (protocols), and constraints. This includes tasks like systems analysis, tradeoff analysis, modeling, simulation, case studies, etc.
Take the subjective average of the below characteristics to derive a rating:

Characteristics	Very low	Low	Nominal	High	Very High	Extra High
Degree of Understanding*	Poor understanding of software architecture and NDI, no documentation	Minimal understanding of architecture and NDI, high-level architectural view	Reasonable understanding of architecture and NDI, some architectural views expressed, e.g. physical & logical views	Good understanding of architecture and NDI, most architectural views expressed	Strong understanding of architecture and NDI, detailed architectural views expressed	Full understanding of architecture, familiar system and NDI, fully documented and maintained architectural views
Percent of required top software architects available to project	Little (20%)	Some (40%)	Often (60%)	Generally (75%)	Mostly (90%)	Full (100%)

*The degree of architectural understanding depends in part on the 4+1 view model described in “Architectural Blueprints – The 4+1 View Model of Software Architecture” or an equivalent set of architectural descriptions.

Software Component Information

- Please describe Software Components
- The word "Component" is used to generically describe a functional unit of software. If the unit is small, describe that unit.
- If the unit has multiple functions, please describe each.
- Add columns if needed.

Component Details

- Component Name (sanitized)
- Component Description
- Component Magnitude
- Lifecycle Phase
- Development/Maintenance Effort
  - Total Effort (hours)
  - Activities Included in Effort (Specification, Architecting, Low-level Design, Implementation, Integration, Acceptance Testing, Other)
- Component Duration
  - Start & End Date
  - Criteria for Start & End dates

Component-level Cost Drivers

- Impact of Software Failure
- Product Complexity
- Developed for Reusability
- Platform Constraints
- Platform Volatility
- Analyst Capability
- Programmer Capability
- Personnel Continuity
- Application Domain Experience
- Language and Tool Expr
- Platform Experience
- Use of Software Tools
- Multi-site Development
- Automated Analysis
- Peer Reviews
- Execution Testing & Tools

Component Driver Example

Use of Software Tools (TOOL)

This driver rates the use of software development tools using three characteristics.
Take the subjective average of the below characteristics to derive a rating:

Characteristics	Very Low	Low	Nominal	High	Very High	Extra High
Coverage	Basic, text based	Simple, interactive	Syntax checking, CM, repository	Semantics checking, code gen, re-engineering	Verification, sophisticated support	N/A
Integration	Incompatible file formats	Convertible file formats	Std format, common GUI, message broadcasting	Shared repository, compatible process assumptions	High degree of commonality, consistent processes	N/A
Maturity	Simple documentation, pre-release	New tools, updated docs, help available	New tools, online docs, tutorials available	Online user support, User's Group	On-site tech support, User's Group	On market more than 3 years

Component Size

- Implementation Programming Language
- Functional Size Measure
  - Description & Count
- SLOC-based Size
  - Baseline Code Count
  - Delivered Code Count
  - New, Adapted, Auto-generated, Deleted Code Counts
  - Adaptation Parameters

- Universal Code Count (UCC) tool outputs are desired

Contact Information

For more information, contact:
Brad Clark
Clarkbk@usc.edu
Or
Brad@software-metrics.com
703-402-3576

Invitation to Participate

- CSSE invites you to collaborate on model development
  - Review model formulation
  - Submit data for model calibration
    - Effort
    - Schedule
    - Model Parameters
    - Actual Size
    - Defects
  - If you contribute data for model calibration, you will receive:
    - An advanced copy of the new model
    - Comparison of your data with respect to other data points used to calibrate the model
- Please talk with me afterwards if you are interested