Socratic Method of Project Characterization

Tim Kurtz
Science Applications International Corporation/NASA Glenn Research Center
Email: Tim.Kurtz@grc.nasa.gov

Abstract

Discussion of ongoing efforts at NASA Glenn Research Center (GRC) to develop a tool for software project characterization utilizing estimation and software control level criteria. The marriage of development cost factors with complexity and risk factors provides a basis for further project characterizations utilizing additional areas of interest with a minimum of additional effort. Characterization results can be used to assist various groups, in planning their development activities. The characterization process consists of a session of answering the questions. As questions are answered, additional questions may be triggered. Once all pertinent questions have been answered, various reports and plans can be generated. The factors can be adjusted to maximize resources, minimize the development efforts and associated controls. The tool is constructed in such a way that organizational processes and tailoring can be added to the resulting reports as well as adding additional characterization questions and answers.

1. Introduction

The Socratic Method consists of five steps 1) posing a question; 2) suggesting a plausible answer or hypothesis; 3) testing the hypothesis; 4) accepting the hypothesis; and 5) acting accordingly. This method allows for modification and revisiting earlier decisions throughout each step.

The Ask Pete tool being developed at GRC embodies the Socratic Method in its project characterization process where the overall question "How can we best develop this software?" is posed. In order to arrive at the ultimate answer, the process of developing the software is broken down into a set of questions related to development effort and software control levels. Answers are suggested and selected by the project manager. These answers may lead to more in-depth questioning until finally a solution is determined. This process results in a characterization of the software project's effort, risks, activities and controls. As with the Socratic Method, once the overall question of how the software can be developed is answered, each of the individual questions can be revisited to gauge their overall affect on the development.

2. Posing the Question

The ultimate question when planning a development effort is "How can we best develop this software?" Ask Pete incorporates COCOMO II factors to determine cost and schedule estimates for the project, the size of the project, and the affect of reuse. GRC has developed a related set of control level factors that help determine the difficulty and the affect success or failure of the project can have on the organization. Once the control level is determined, a reasonable set of development activities is defined to facilitate success of the project.

2.1 Development Effort Questions

COCOMO has been used for twenty years by corporations and DoD to estimate software development projects. COCOMO II also does a good job of characterizing a project with the information it needs to provide the estimate. It also coaches the project manager to look into the different aspects of a project; to think about the choices that are available; and forces him to consider various factors and drivers.

The COCOMO II equation:

$$PM_{NS} = A \times Size^{E} \times \prod_{i=1}^{n} EM_i$$

where:

PM_{NS} = nominal schedule effort in person-months

A = 2.94

Size = code estimate (thousand Source Lines Of Code, KSLOC, or Unadjusted Function Point, UFP)

EM = Effort Multipliers

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j$$

B = 0.91

SF = Scale Factors

COCOMO II Scale Factors (ref. Table 1) and Effort Multipliers (ref. Table 2) are used to determine the cost and schedule estimates for the project, the size of the project, the need for reuse. These factors also assist in defining the personnel resource requirements available to the project, performance and schedule constraints, and

Scale Factors	Very low	Low	Nominal	High	Very high	Extra High
Precedentedness						
Organizational understanding of product objectives	Thoroughly unprecedented	Largely unprecedented	Somewhat unprecedented	Generally familiar	Largely familiar	Thoroughly familiar
Experience working with related systems;						
Concurrent development						
Innovative architectures and algorithms						
Development Flexibility						
Conformance to pre-established requirements	Rigorous	Occasional relaxation	Some relaxation	General conformity	Some conformity	General goals
Conformance to external interfaces						
Premium on early completion						
Architecture and Risk Resolution						
Risk management planning (RMP)	Little (20%)	Some (40%)	Often (60%)	Generally (75%)	Mostly (90%)	Full (100%)
Schedule, budget and milestones compatible with RMP						
Schedule devoted to architecture;						
Architects available						
Risk and architecture tool support available						
Uncertainty in architecture, mission, user interface, etc.						
Risk items						
Team Cohesion						
Stakeholder Objectives and cultures	Very difficult interactions	Some difficult interactions	Basically cooperative interactions	Largely cooperative	Highly cooperative	Seamless interactions
Willingness to accommodate others' objectives						
Experience operating as a team						
shared vision and commitments						
Capability Maturity Model						
Process maturity level	Level1 (bottom half)	Level 1 (top half)	Level 2	Level 3	Level 4	Level 5

Table 1, Scale Factors

safety risks. As a whole, these questions provide an indication of the length of the development effort, the personnel resources required and the cost.

Scale Factors account for the comparative economies or diseconomies of scale associated with software development projects of various sizes. If $E < 1.0$ then the project shows economies of scale and as the size of the project increases, the effort to complete the project increases at a lesser rate. If $E = 1.0$ which is normal for small projects, then the effort increases at a linear rate as the size increases. When $E > 1.0$, the project shows diseconomies of scale. Five factors are evaluated to account for this value

While two types of effort multipliers, Post-architecture Cost Drivers and Early Design Model Driver, can be looked at with the COCOMO II model. The Ask Pete tool incorporates both.

The Early Design Model Drivers evaluate seven areas of the project:

- Product Reliability and Complexity – sum of the combination of reliability and documentation; with product complexity; and data base size
- Developed for Reusability
- Platform Difficulty – sum of the combination of time and storage constraints; with platform volatility
- Personnel Capability – sum of the combination of analyst and programmer capability; with annual personnel turnover
- Personnel Experience – sum of applications; platform; and language and tool experience
- Facilities – sum of tool support and multisite conditions
- Required Development Schedule

	Very low	Low	Nominal	High	Very high	Extra High
Product Factors						
Required Software Reliability – effect of software not performing its intended function over a period of time, measured by amount of inconvenience, losses and risk to life	Slight inconvenience	Low, easily recoverable losses	Moderate easily recoverable losses	High financial loss	Risk to life	
Database Size – effect of test data requirements on testing the software (testing DB bytes/pgm SLOC)		< 10	10 to 100	100 to 1000	> 1000	
Product Complexity – average of the various types of operations that characterize the software. They are control; computational; device-dependent; data management and user interface management.	Simple input forms, simple arrays, etc.	Simple GUI, single file, etc.	Widget set, multi-file input, etc.	Widget set development, complex data structuring	Dynamic graphics, distributed database, etc.	Complex multimedia, relational/object structures.
Developed for Reusability – effect of constraints imposed by developing software to be reusable		None	Across project	Across program	Across product line	Across mult product lines
Documentation Match to Life-Cycle Needs – effect of generating documentation on the development effort	Many lifecycle needs not covered	Some lifecycle needs not covered	Satisfactory for lifecycle	Excessive for lifecycle needs	Very excessive for lifecycle needs	
Platform Factors						
Execution Time Constraint – the percent of execution time expected to be available			50% or less	70%	85%	95%
Main Storage Constraint – the percent of storage expected to be available						
Platform Volatility – the expected frequency and magnitude of changes to the target machine hardware and infrastructure software		Major – 12 months, minor – 1 month	Major – 6 months, minor – 2 weeks	Major – 2 months, minor – 1 week	Major – 2 weeks, minor – 2 days	
Personnel Factors						
Analyst Capability – rate, as a team, the personnel who work on requirements and design for analysis and design ability, efficiency and thoroughness, and ability to communicate and cooperate	15th percentile	35th percentile	55th percentile	75th percentile	90th percentile	
Programmer Capability – rate, as a team, the programmers for ability, efficiency, thoroughness, and ability to communicate and cooperate.						
Personnel Continuity – annual personnel turnover	48%	24%	12%	6%	3%	
Applications Experience – rate the project team's experience developing this type of application	2 months or less	6 months	1 year	3 years	6 years	
Platform Experience – rate, as a team, the personnel experience working with the platform including more powerful capabilities, such as, graphical user interfaces, databases, networking, etc.						
Language and Tool Experience – rate, as a team, the overall experience in working with the programming language and software tools..						
Team Cohesion						
Use of Software Tools – evaluates the availability and type of software tools	Edit, code, debug	Simple frontend, backend CASE, little integration	Basic lifecycle tools, moderately integrated	Strong, mature lifecycle tools, moderately integrated	Strong, mature proactive lifecycle tools, well integrated with processes	
Multisite Development – location of development teams and communication support	International	Muilti-city and multi-company	Multi-city or multi-company	Same city or metro area	Same building or complex	
	Some phone, mail	Phone, FAX	Email	Electronic comm.	Elect comm., video confe	
Required Development Schedule – schedule constraint imposed on the development effort	75% nominal	85% nominal	Nominal	130% nominal	160% nominal	
Capability Maturity Model						
Process maturity level	Level1 (bottom half)	Level 1 (top half)	Level 2	Level 3	Level 4	Level 5

Table 2, Effort Multipliers

Factor	1	2	3	4	5
Resourcing					
Software cost annualized over life of project (including all civil servants and contractors)	$100K - $500K	$500K – 1M	$1M - $2M	$2M - $20M	$20M and up
Organizational Complexity					
Project development location	Own Group	Several Groups, most at GRC	More than 2 other sites	More than 3 other sites	Numerous sites
Customers	Self	Other in own Directorate or one customer group, low number of users	Within GRC	Within NASA	Entire Industry or multiple industries
Developers Software experience level	All team qualified and experienced	Most team members qualified and experienced	Half the team qualified and experienced	Few team members experienced	Experienced staff not available
Technical Complexity					
Test Risk	No testing required	Minimum Testing	Standard testing required	Integrated Testing	Major testing effort required (e.g. IV&V)
Degree of Innovation	Well proven, known to GRC	Proven with some GRC experience	Proven, but new to GRC	Partially proven with some pioneering	Pioneering
Software development tool availability	All software development tools already purchased/in-house/familiar with	Majority of the software development tools are purchased /in-house/familiar with	Software development tools must be identified and purchased and learned	Majority of software development tools must be obtained, remainder developed	All software development tools must be developed
Interdependencies of deliverables	Simple standalone	Some Integration	Integrated	Highly integrated	Fully integrated
Safety Implications * (see NASA STD 8719.13A, NASA GB-1740.13 & NHB 1700.1 (V1-B))					
Potential Damage to carrier vehicle, major equipment, or system itself	No damage	Small/minor damage to equipment, or to system itself, mission still possible	Repairable/ recoverable damage to system and little or no damage to any related or surrounding systems	**Loss of system, and/ or damage to any critical surrounding systems or carrier vehicle**	**Loss of carrier vehicle (e.g. space craft, aircraft, major satellite)**
Potential Injury to personnel	No injury	Minor injury	Injury	**Severe injury or temporary disability**	**Loss of life or permanent disability**
Business Implications					
Consequence of Failure	Minor loss of Customer Confidence	Unsatisfied Customer	Damage to GRC Reputation	Damage to NASA reputation	**Significant impact to USA**
Schedule Pressure	No time pressure	Little effort to meet milestones	Nominal effort to meet milestones	Aggressive effort to meet milestones	Time critical

Critical Control

|----------Low control software ----------|

|---------- Medium control software----------|

|---------- High control software ----------|

Table 3, Control Level Factors

Post Architecture Cost Drivers look at the four main areas of a project, the product itself, the platform it will operate on, the development personnel and the project constraints. Some of these factors can be influenced by the project manager and some can not:

- Product - characterize the software being developed. The higher these factors are, the more effort will be needed to accommodate them.
- Platform - characterize the target machine (hardware and infrastructure software) which the software will eventually operate on. The higher

these factors are, the more effort will be needed to accommodate them.

- Personnel - have a large effect on determining the amount of effort needed to develop software. The more capability and experience personnel on the development team have, the greater the effect on reducing the development estimate.
- Project - characterize the influences on the project from availability of tools, development location and schedule

2.2. Software Control Level Questions

Control level questions are used to identify the challenges to successful development. They address the affect of the organizational and technical complexities, and the consequence of failure of the project – Who are the customers (internal or external)? What amount of technical complexity is involved (the degree of innovation, use of tools, interdependency of the deliverables)? What is the consequence of failure (safety implications, business implications, lost resources)? What is the risk to the organization, the project, the users and associated equipment?

As with the development effort questions, control level questions are bounded with a fixed set of possible answers (ref. Table 3), each of which is assigned a value. The resulting score determines whether the project falls into the Low, Medium, High or Critical control level. A generic lifecycle, based on the waterfall model is associated with each of the control levels.

As shown in Figure 1, High and Critical control level projects follow the typical lifecycle with User and Software Requirements, Design, Development (Code and Unit Test), and System Test phases. Medium control level projects utilize a modified lifecycle with the Software portion of the Requirements phase included in the Design phase and the System Testing phase combined with the Development phase. Low control level projects, follow a simplified lifecycle consisting of User Requirements, Development, which includes Design and System Testing phases.

Typically, Critical and High Control level projects include formal inspections of code and documentation, intensive Software Assurance involvement, IV&V and substantial documentation requirements. Critical control level projects also have increased emphasis on safety factors. Medium control level projects have moderate Software Assurance involvement and a lesser amount of documentation for operational concept and post release activities. Low control level projects have minimal Software Assurance activity and much of the planning is combined into a single management or development plan. As an example, Table 4 contains a listing of all possible activities for the Requirements phase and their applicability by control level.

Requirements Phase Activities	Low	Medium	High	Critical
Authorization to proceed	☒	☒	☒	☒
Identify design/coding standards	☒	☒	☒	☒
Management Plan approval	☒	☒	☒	☒
Documented requirements	☒	☒	☒	☒
Requirements approval	☒	☒	☒	☒
Maintain Software Development Folder	☐	☒	☒	☒
Software Assurance reviews Management Plan	☐	☒	☒	☒
Implement Problem report and corrective action system	☐	☒	☒	☒
Peer review of requirements	☐	☒	☒	☒
Software Assurance reviews requirements	☐	☐	☒	☒
Peer review of plans	☐	☐	☒	☒
Implement Formal configuration management	☐	☐	☒	☒
Conduct Product Assurance Audits	☐	☐	☒	☒
Conduct Formal Reviews	☐	☐	☒	☒
Document approval of requirements and formal review	☐	☐	☒	☒
Conduct formal inspection of requirements	☐	☐	☐	☒
Customer approval of certification procedures	☐	☐	☐	☒
Conduct analyses of criticality and safety	☐	☐	☐	☒
Plan and schedule IV&V activities	☐	☐	☐	☒
Identify method for verification of safety critical functions and requirements	☐	☐	☐	☒

Table 4, Requirements Phase Activities by Control Level

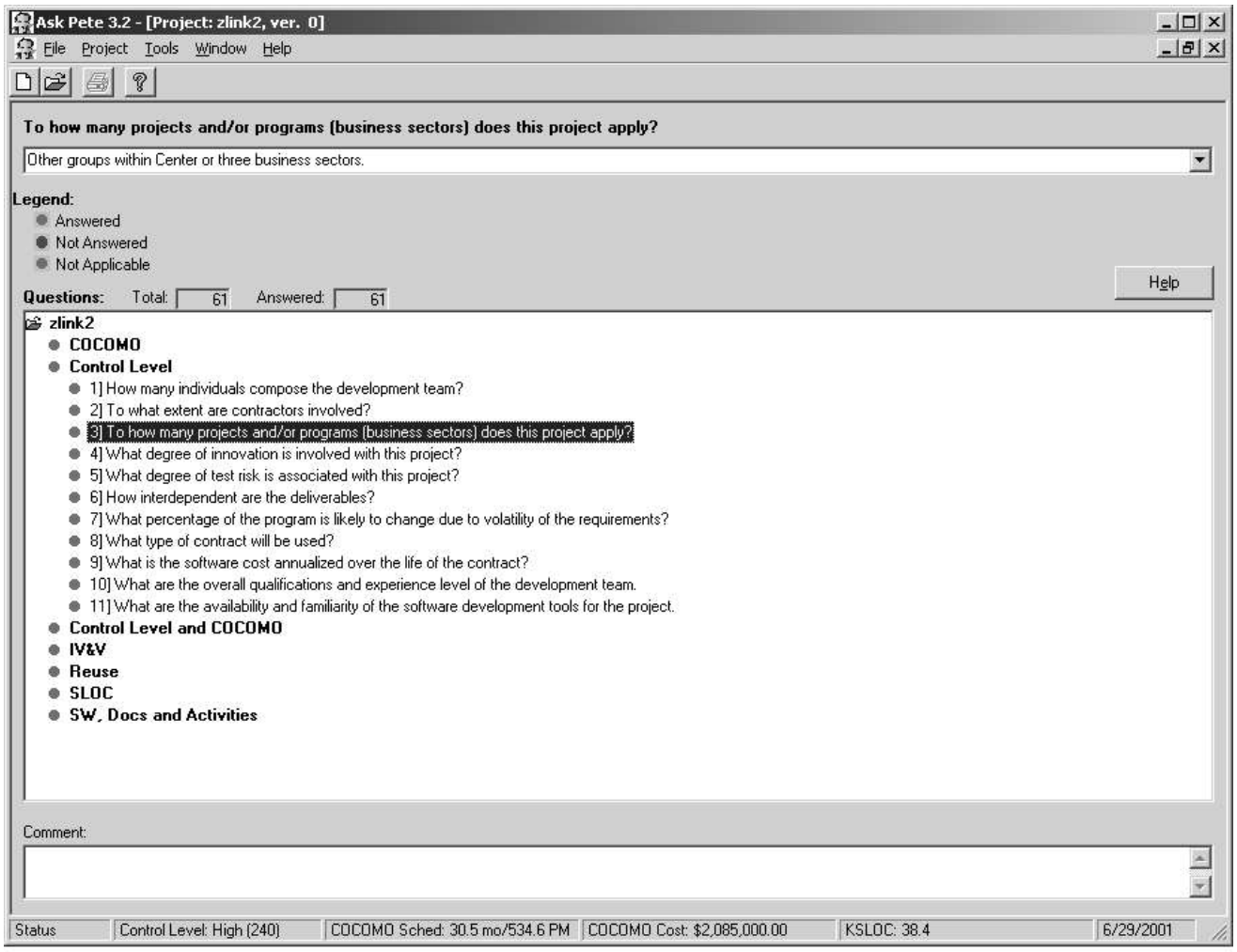

Figure 2, Completed Project

3. Suggesting a Plausible Answer

Initially, the tool should be used at the inception of the project and all the questions answered based upon what is known or expected at that time. This will provide an initial answer to the question of how the software can best be developed, without being colored by cost, schedule or resource issues. Additionally, it will serve as a baseline for comparing future changes and identify areas where more information is needed to complete the planning effort.

The following reports can now be generated by Ask Pete:

- Results report – containing the calculated values for each COCOMO II factor, the control level and associated activities and documentation requirements, COCOMO II cost and schedule estimates, including personnel requirements for each phase.
- Software Development Plan – tailored to the control level
- Software Assurance Plan – tailored to the control level with Activities and associated time estimates for each phase

4. Testing the Hypothesis

Typically, the project manager will want to reduce the control level, development cost, and development schedule. Once all questions have been answered and the initial planning has been completed, the Control level, cost and schedule estimates are displayed in the status bar (ref. Figure 2). As responses to the questions are changed, these values are updated allowing the project manager to see the effect of the changes on the overall project plan.

Care should be exercised to ensure that the changes made are within the scope of the project manager's ability to control since the results generated by Ask Pete are

dependent on the responses selected. Selecting "Simple, standalone application" as the response to the question, "How interdependent are the deliverables?" may reduce the control level of his project from High to Medium. If the software will in fact be heavily integrated with other deliverables, the project manager may find that inadequate resources are recommended for integration, resulting in failure of the project to be delivered on time.

Project managers should strive to maximize their resources and minimize the development effort using the tool. They should also realize that in order for their project to be successful the decisions they make will affect the resulting plans produced by Ask Pete.

5. Accepting the Hypothesis

Once the project has been tuned to the best of the project manager's ability, two courses of action are available:

1. If the results are satisfactory then the plans and schedules should be generated and fine tuned for the project, i.e. identifying project personnel and resources, finalizing the schedule, etc. Once this has been completed, the plans and schedules should be distributed to all associated project personnel for approvals and implementation.
2. If the results are not satisfactory, i.e. control level is too high or cost is not within the allowable budget, etc., then the decision must be made whether to abandon the project or negotiate for a combination of reduction in requirements and increase in resources and schedule. In any case, if the project is not abandoned, the earlier characterization must be revisited.

6. Act Accordingly

Finally, in order for the project to be successful, the resulting plan must be followed.

During the life of the project changes will no doubt occur, i.e. increase/decrease in scope, change in available funding/personnel or resources, etc. As these occur, the characterization process should be repeated to determine if the changes affect the current plan or the viability of the project. If they do then the appropriate actions should be taken.

7. Tool Adaptability

Ask Pete uses an Access database that holds the questions, answers or facts and the decision processing which generates the report based on a project's inputs. The database includes entry forms that allow for addition, deletion and modification of the questions, answers and outputs. This allows the tool to be easily adapted to any organization's development methodology.

During the development of Ask Pete, the development team was requested to incorporate the NASA IV&V criteria and add IV&V recommendations to the tool's reports. The IV&V determination process consists of a matrix of 17 questions and answers with in a format similar to the control level questions.

We found that the characterization process using COCOMO II and the control level factors had already addressed all but three of the 17 IV&V questions. These questions and the allowable responses were added to the database along with the resulting report outputs and decision processing. This indicates that the COCOMO II and control level factors provide an excellent base for enhancing software project characterization and obtaining additional information.

8. Additional Work

Ask Pete has also been integrated with the Advanced Risk Reduction Tool (ARRT) being developed by Dr. Martin Feather at NASA Jet Propulsion Laboratory. ARRT provides a risk balancing capability for software projects using the Software Engineering Institute's Software Risk Taxonomy as a base set of failure modes and Ask Pete's software development activities as risk mitigation activities for the failure modes.

When a project is saved in Ask Pete, various project information is exported to an Access database that is available to ARRT and other applications. The database includes:

- Project estimates
- Control level
- The project's development activities

Ask Pete is available for download at http://tkurtz.grc.nasa.gov/pete.

9. Acknowledgements

The research described in this paper was carried out at NASA Glenn Research Center under contracts with the National Aeronautics and Space Administration. Funding has been provided through the NASA Office of Safety and Mission Assurance under the NASA Software Program lead by the NASA Software IV&V Facility, UPN 323-08-5P. The author gratefully acknowledges the significant contributions, guidance and assistance of Martin Feather, Tim Menzies, Marcus Fisher, Frank Huy, Kenneth McGill, and Siamak Yassini, with special appreciation to Martha Wetherholt.

10. References

[1] T. Kurtz, "AskPete Web site" http://tkurtz.grc.nasa.gov/pete

[2] T. Kurtz and M.S. Feather, “Putting it All Together: Software Planning, Estimating and Assessment for a Successful Project”, in Proc. of 4th Int. Software Quality & Internet Quality Conf., Brussels, Belgium, Nov. 2000.

[3] T. Menzies and E. Sinsel and T. Kurtz, “Learning to Reduce Risks with COCOMO-II”, Workshop on Intelligent Software Engineering, an ICSE 2000 workshop, and NASA/WVU Software Research Lab, Fairmont, WV, Tech report # NASA-IVV-99-027, 1999.

[4] B. Boehm, et al., 2001, *Software Cost Estimation with COCOMO II*, Prentice Hall, Upper Saddle River, NJ, 2001