Criteria for Selecting Software Process Models

Linda C. Alexander[1]
Center for Software Engineering
U.S. Army Communications-Electronics Command
Fort Monmouth, NJ 07703

Alan M. Davis[2]
Center for Software Systems Engineering
George Mason University
Fairfax, VA 22030

Abstract

In an effort to improve the quality of software and its production, researchers have defined many variations of the software development process. These range from the traditional waterfall process to the throwaway prototype. This paper presents guidelines for selecting the most appropriate process model for a particular project. To facilitate selection, we have organized the software process models into a three-level hierarchy. After applying all the criteria a relative rating of the process models is generated to guide the selection.

1 The Process Model Selection Problem

Many factors influence the success of a software project. These include the skill of the software developers, the ability of the project managers, the availability of CASE tools, the computing environment, the difficulty of the problem being solved, the resources available, and the process model being employed. In the last category, there is a plethora of process models to chose from [1].

The selection of an inappropriate process model can result in (1) a system that does not satisfy user needs, (2) increased cost, and (3) longer development times. For example, if user needs are poorly understood or changing, the use of a conventional process model could result in a system that does not satisfy the user's needs. On the contrary, if user needs are well understood and stable, then building a prototype can unnecessarily increase the development cost and schedule [8].

Today's project managers choose a software process model using ad hoc, unjustified, and often undocumented criteria. Boehm's spiral model recognizes the need for process model selection and bases the selection on just one criterion: risk [7]. Other authors have also identified this need [4]. It was not until 1990 that any attempt was made to generate a comprehensive set of criteria and values for such selections [2]. This paper describes 20 criteria and a method for applying them. Using this method, the project manager evaluates a project with respect to the criteria to develop a prioritized list of process models. We will use the following notations:

- C is the set of twenty criteria with elements c_i, i = 1..20 (e.g., c_1 is *user experience*, c_8 is *frequency of changes*);
- V_i is the ordered vector of elements v_{ij} of values that each criterion c_i may take on (e.g., v_{11}, v_{12} and v_{13} have values *novice*, *experienced*, and *expert*, respectively; while v_{81}, v_{82} and v_{83} have values *seldom*, *slow* and *rapid*);
- s_{ij} is a binary indicator of which value of criterion c_i applies to this particular project (e.g., s_{11}, s_{12}, s_{13}, s_{81}, s_{82}, and s_{83} are 0, 0, 1, 1, 0, and 0 respectively). The s_{ij}'s form a project characteristics matrix that reflects the attributes of a project; and
- a_{ij} represents the applicability of a process model to a v_{ij} (e.g., a_{13} = 1 for the conventional process model indicates that process is appropriate for projects having *expert* users). The a_{ij}'s for each process model form a matrix used to determine the rating of a process model with respect to a particular project.

This paper supplies the set of criteria, C, their possible values, V_i's, and a process model matrix for each process model containing the a_{ij}'s. The s_{ij}'s are project dependent and are determined by the project manager.

To determine the appropriate process model for a project, the project manager follows a three step method.

1. Examine the project's attributes and determine s_{ij}'s for each criterion that best describe the project.

[1]This research was conducted while attending George Mason University.

[2]This research supported in part by U.S. Army Communications-Electronics Command and COMPASS Corporation under contract #DAAB07-88-B029, and by Commonwealth of Virginia Center for Innovative Technology under grant #STC-87-002.

2. For each process model, compute:

$$RATING = \sum_{i=1}^{20} \sum_{j=1}^{3} (s_{ij} * a_{ij})$$

3. The process model with the highest *RATING* is the best choice, second highest is the second best choice, etc.

In general, software development process models define the partitioning, ordering, and temporal relationships that exist among the activities of software development. However, not all process models address these at the same level of abstraction. Thus, some software process models delineate broad groups of activities over the software life cycle, while others are detailed to the point of specifying the particular methods and tools used to perform the prescribed activities. The method described above must be applied at each level of abstraction to select the appropriate process model at that level.

Section 2 of this paper describes the process selection criteria (C) and the ranges of values for these criteria (V_i's). Section 3 describes some process models in use today and organizes them into a hierarchy. Section 4 establishes the correspondence between each criterion value and each process model (a_{ij}'s). Section 5 provides a short example. Section 6 summarizes our conclusions.

2 Initial Project Criteria

The project criteria (C) fall into the following general categories: personnel, problem, product, resource, and organizational. Table 1 shows the range of values (V_i) for each of the criterion (c_i).

2.1 Personnel Criteria

The personnel criteria concern both the developers and users. The personnel criteria and their values are described below.

- Users' Experience in Application Domain (c_1): This criterion assesses the users' knowledge of the domain of the problem. Values are: V_1 = (*novice, experienced, expert*). Each end of this scale can work both for and against an effort. Novice users may not know much about the problem area, but they will be more accepting of new or different solutions. Expert users will know a lot about the problem, but may be resistant to changes.
- Users' Ability to Express Requirements (c_2): This criterion evaluates how well the users can communicate their needs. Values are: V_2 = (*silent, communicative, expressive*). At the silent end of the scale are users who can't say what they need, but will recognize it when they see it. Expressive users are able to say exactly what they think they want.
- Developers' Experience in Application Domain (c_3): This criterion assesses the developers' knowledge of the problem domain. The developers' knowledge can be the result of being a user in the application domain, or developing other applications in the domain. Values are: V_3 = (*novice, experienced, expert*).
- Developers' Software Engineering Experience (c_4): This criterion evaluates the developers' experience with and knowledge of the software tools, methods, techniques, and languages needed for a development effort. Values are: V_4 = (*novice, experienced, expert*).

2.2 Problem Criteria

The problem criteria concern the problem being solved by the software development effort. The problem criteria and their values are discussed below.

- Maturity of the Application (c_5): This criterion evaluates the amount of general knowledge of the problem. Development of software in mature application areas can benefit from previous development efforts, while there is less background knowledge in newer application domains. Values are: V_5 = (*new, standard, well-established*). An example of a well-established application area is payroll. Much artificial intelligence software represents a new application.
- Problem Complexity (c_6): This criterion measures the complexity of the problem to be solved. Values are: V_6 = (*simple, difficult, complex*). Large, real-time, parallel systems tend to be complex, while smaller, sequential systems are usually simple.
- Requirement for Partial Functionality (c_7): This criterion measures the practicality and/or need to deliver intermediate products that provide only a part of the eventual full functionality of the target product. Values are: V_7 = (*urgent, desirable, not desirable*). The life support system aboard a space shuttle is an example where partial functionality is not desirable. The automation of office functions is an example where early delivery of the word processing functions might be desirable.
- Frequency of Changes (c_8): This criterion gauges the frequency at which the problem changes. Values are: V_8 = (*rapid, slow, seldom*). An example that changes rapidly is a weapons system because it must respond to the rapidly changing threat environment. A compiler for a standard language is an example of a more slowly changing problem.

Table 1. Selection Criteria and Values

c_i	CRITERIA	v_{i1}	v_{i2}	v_{i3}
c_1	User's Experience	Novice	Experienced	Expert
c_2	User's Expression	Silent	Communicative	Expressive
c_3	Dev's Applic Exper	Novice	Experienced	Expert
c_4	Dev's Sw Experience	Novice	Experienced	Expert
c_5	Mature Of Application	New	Standard	Establish
c_6	Problem Complexity	Simple	Difficult	Complex
c_7	Partial Functionality	Not Desire	Desirable	Urgent
c_8	Frequency Of Changes	Seldom	Slow	Rapid
c_9	Magnitude Of Changes	Minor	Moderate	Extreme
c_{10}	Product Size	Small	Medium	Large
c_{11}	Product Complexity	Simple	Difficult	Complex
c_{12}	"Ility" Requirements	Flexible	Moderate	Exacting
c_{13}	Interface Rqmts	Minor	Significant	Critical
c_{14}	Funds Profile	Low-high	Level	High-low
c_{15}	Funds Availability	Scarce	Adequate	Ample
c_{16}	Staff Profile	Low-high	Level	High-low
c_{17}	Staff Availability	Scarce	Adequate	Ample
c_{18}	Access To Users	None	Limited	Free
c_{19}	Mgmt Compatibility	Guideline	Flexible	Enforced
c_{20}	QA/CM Compatibility	Basic	Intermed	Advanced

- Magnitude of Changes (c_9): This criterion assesses the relative size of expected changes in the problem. Values are: V_9 = (*extreme, moderate, minor*). A new communications protocol may result in a small change to a telephone system. The fielding of a new sensor may require a complete redesign of a defense system.

2.3 Product Criteria

The product criteria concern the actual software product to be developed. The product criteria and their values are discussed below.

- Product Size (c_{10}): This criterion measures the expected size of the final product. Since this measurement is being done at the start of the development effort, it is only an estimate. Using Fairley's [11] product sizing definitions (shown in parentheses), the values are: V_{10} = (*large* (very large and extremely large), *medium* (medium and large), *small* (trivial and small)).
- Product Complexity (c_{11}): This criterion gauges the complexity of the software to be developed. Values are: V_{11} = (*complicated, difficult, simple*).
- "ility" Requirements (c_{12}): The "ilities" represent the non-behavioral requirements placed on a software product, such as reliability, adaptability, provability, usability, reusability, and maintainability. This criterion measures the relative burden of such requirements. Values are: V_{12} = (*exacting, moderate, flexible*). Software whose failure could jeopardize life is an example with exacting reliability and provability requirements. Software that is expected to be in use for a long time might have exacting adaptability and maintainability requirements.
- Human Interface Requirements (c_{13}): This criterion measures the criticality of the human interface. Values are: V_{13} = (*critical, significant, minor*).

2.4 Resource Criteria

Resource criteria concern resources available for development. Unlike other criteria, which are evaluated by selecting a value on a scale, funding and staffing criteria are evaluated by the profile of the resource over time. Resource criteria and their evaluation are discussed below.

- Funding Profile (c_{14}): This criterion measures the amount and availability of funds for the development effort. The graphs shown in Figure 1 represent the general shapes of the majority of project funding profiles. Profile (a) Low-High shows the amount of funds increasing at the beginning of the effort, with the largest amount of funds available near the end. Profile (b) High-Low shows the majority of funds available at the beginning of the effort. Profile (c) Level indicates a steady level of funding over the effort. Thus, V_{14} = (*Low-High, High-Low, Level*).
- Funds Availability (c_{15}): This criterion measures the adequacy of the funds available for an effort. Values are: V_{15} = (*scarce, adequate, ample*).
- Staffing Profile (c_{16}): This criterion assesses the numbers of people available over time for the effort. Profiles for staffing are the same as for funding. As

shown in Figure 1, V_{16} = (*Low-High, High-Low, Level*).

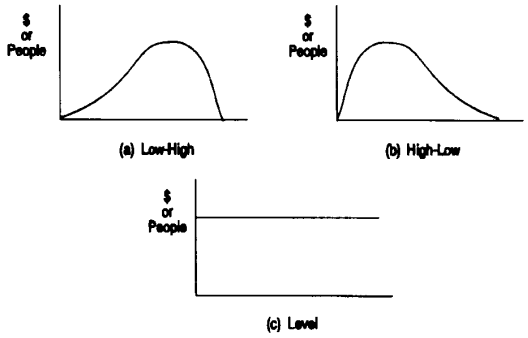

Figure 1. Funding and Staffing Profiles

- Staff Availability (c_{17}): This criterion gauges the sufficiency of the staff available for a project. Values are: V_{17} = (*scarce, adequate, ample*).
- Accessibility of Users (c_{18}): This criterion measures the amount of access developers have to users. Values are: V_{18} = (*no access, limited access, free access*).

2.5 Organizational Criteria

Organizational criteria relate to how organizational policies impact a development effort. The organizational criteria are discussed below.

- Management Compatibility (c_{19}): This criterion measures the degree to which a software process model is compatible with the organization wide development requirements. Values are: V_{19} = (*strictly enforced, flexible, guideline only*).
- Quality Assurance and Configuration Management Capability (c_{20}): This criterion assesses the compatibility between a particular process model and the organization's quality assurance and configuration management procedures. Values are: V_{20} = (*basic, intermediate, advanced*).

3 A Hierarchy of Process Models

The variety of process models in use today is staggering, with the level of abstraction at which each treats the software process being a significant variant. The top level of the hierarchy contains process models that delineate broad groups of activities. The next level contains process models that further refine the grouping and relationships among the prescribed activities. At the next level are the process models that specify particular tools and methods for performing activities. Figure 2 shows one example of this hierarchy. In this figure, the top level (level 3) shows a particular process model: determine problem, development, and operations and maintenance, performed sequentially. Level 2 shows a particular process for performing development. Level 1 shows a particular set of tools and techniques to perform the requirements activities of level 2. The following sections describe some of the process models at levels 3 and 2. Level 1 methods and tools are beyond the scope of this paper. Selection of level 1 methods and tools for requirements has been discussed in [9].

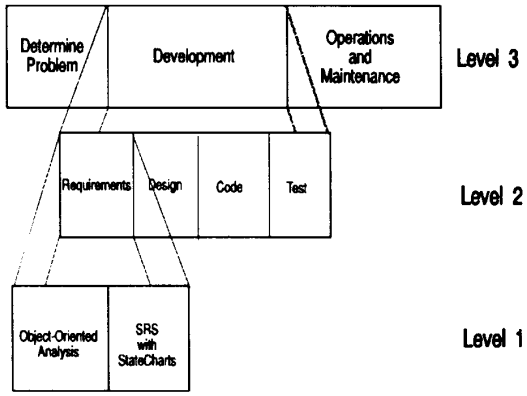

Figure 2. The Hierarchy of Software Process Models

3.1 Level 3 Process Models

Level 3 process models attach labels to segments of time to characterize activities taking place. These models also address any partitioning of user needs. This work will consider three examples: conventional, incremental, and evolutionary.

Under the *conventional* software process model the problem is not subdivided. A product is developed to address the entire problem. The software life cycle is divided into three segments: determine the problem to be solved, develop the product, and operate and maintain. See Figure 3.

Figure 3. The Conventional Software Process Model

Under the *incremental* software process model there is a conscious decision to delay development of a portion of the target software product [5]. The entire problem is subsetted before any interim products are developed. The product of each interim development will be used in the operational environment, and will most likely not have any maintenance effort applied to it. Figure 4 shows the incremental software process model.

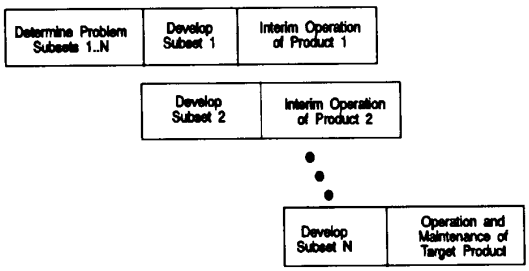

Figure 4. The Incremental Software Process Model

The *evolutionary* software process model is similar to incremental in that there is a conscious decision to delay development of a portion of the target software product [12]. Unlike incremental development, however, interim products are developed before the next subset is determined. See Figure 5.

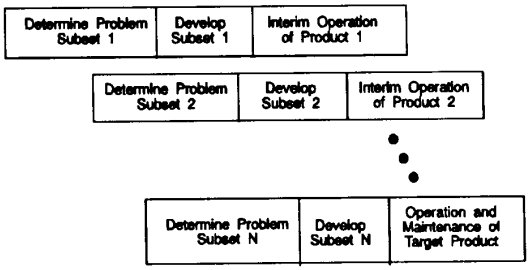

Figure 5. The Evolutionary Software Process Model

3.2 Level 2 Process Models

Level 2 process models specify activities to be performed in each of the time segments of a level 3 process model. These models also address the different representations of the product that will be constructed. This work will consider three examples: waterfall, hybrid prototyping, and operational specification.

The *waterfall* model was first introduced by Royce in 1970 [15]. Figure 6 shows a typical waterfall software process model [6].

In *hybrid prototyping*, prototypes are constructed initially. At later stages, parts of the prototypes may be selectively discarded, while other parts (e.g., objects, reusable code) are incorporated into the product (Luqi uses the term rapid prototyping, but this can also refer to building throwaway prototypes quickly [14]). This should not be confused with throwaway prototyping [13] which is often used as a method in a level 1 process model to perform some of the prescribed activities. Figure 7 shows such a level 2 software process model.

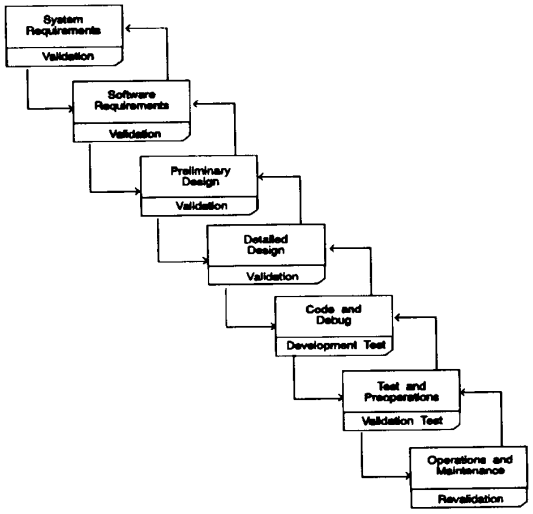

Figure 6. The Waterfall Software Process Model
© 1976 IEEE

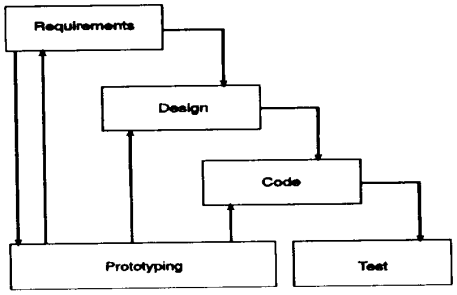

Figure 7. The Hybrid Prototyping Process Model

Operational specification supplies an executable representation of the product directly from specifications. See Figure 8 [1]. The first phase describes the external behavior of a solution system. Subsequent phases transform it into a more robust, operational system.

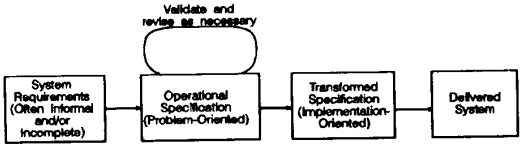

Figure 8. The Operational Specification Process Model
© 1986 IEEE

3.3 Summary

The process models presented above are not the only process models in existence. For example, operational

prototyping [10] and concurrent software modeling [16] offer alternative level 3 models. The transformational process model [3] is an alternative level 2 model.

4 Process Model Selection Tables

This section contains tables indicating the appropriateness of a process model to projects exhibiting particular values v_{ij} for criteria c_i. These were developed by looking at features of process models and features needed in particular project situations. The key features considered were:

1) the opportunity to modify, validate, and/or verify the software;
2) the degree of functionality delivered and when; and
3) the level of activity with respect to time.

An entry of 1 in a table indicates that the process model is appropriate for projects exhibiting that value of a criterion. An entry of 0 indicates that the process model is inappropriate. Thus Table (?) shows that the conventional process model is appropriate for projects with *expert* users, *expressive* users, *established* applications, etc. Tables (?) through (?) display the same information for the remaining process models.

5 Example

This section contains an example of using the process model matrices. The example was developed from an actual project, the Prototype Ocean Surveillance Terminal (POST) [10], with the project characteristics determined by a member of the development team.

Step 1: Examine the project and determine s_{ij}'s for each criterion c_i. The objective of this project was to place desktop workstations aboard ships to analyze strategic intelligence information. The application was new to both users and developers and involved a complex, rapidly changing problem. There was a need for quick delivery of a product with any capability. The user interface represented a critical part of the requirements. Table 8 captures the characteristics of this project by specifying the s_{ij}'s.

Step 2: Compute the rating for each process model using the formula:

$$RATING = \sum_{i=1}^{20} \sum_{j=1}^{3} (s_{ij} * a_{ij})$$

Table 9 shows the results of performing this step for the conventional process model. Each entry in the table is the result of the multiplication $(s_{ij} * a_{ij})$. The numbers in the column to the right are the results of the summation over j. The total at the bottom of the right-hand column is the result of the sum over i. For this project the conventional process model received a rating of 8. The results for all of the process models are shown in table 10.

Table 2. Conventional Selection Matrix

Criteria	v_{i1}	v_{i2}	v_{i3}
c_1	0	0	1
c_2	0	0	1
c_3	0	0	1
c_4	1	1	1
c_5	0	0	1
c_6	1	0	0
c_7	1	0	0
c_8	1	0	0
c_9	1	0	0
c_{10}	1	0	0
c_{11}	1	0	0
c_{12}	1	1	1
c_{13}	1	0	0
c_{14}	1	0	0
c_{15}	0	0	1
c_{16}	1	0	0
c_{17}	0	0	1
c_{18}	0	1	1
c_{19}	1	1	1
c_{20}	1	1	1

Table 3. Incremental Selection Matrix

Criteria	v_{i1}	v_{i2}	v_{i3}
c_1	0	1	1
c_2	0	1	1
c_3	0	1	1
c_4	0	1	1
c_5	0	1	1
c_6	1	1	0
c_7	0	1	0
c_8	1	1	0
c_9	1	1	0
c_{10}	1	1	1
c_{11}	1	1	1
c_{12}	1	1	0
c_{13}	1	1	0
c_{14}	0	1	0
c_{15}	0	1	1
c_{16}	0	1	0
c_{17}	0	1	1
c_{18}	0	1	1
c_{19}	1	1	0
c_{20}	0	1	1

Table 4. Evolutionary Selection Matrix

Criteria	v_{i1}	v_{i2}	v_{i3}
c_1	1	1	1
c_2	1	1	1
c_3	1	1	1
c_4	0	1	1
c_5	1	1	1
c_6	1	1	1
c_7	0	1	1
c_8	1	1	1
c_9	1	1	1
c_{10}	1	1	1
c_{11}	1	1	0
c_{12}	1	0	0
c_{13}	1	1	1
c_{14}	0	1	0
c_{15}	1	1	1
c_{16}	0	1	0
c_{17}	1	1	1
c_{18}	0	0	1
c_{19}	1	1	0
c_{20}	0	1	1

Table 5. Waterfall Selection Matrix

Criteria	v_{i1}	v_{i2}	v_{i3}
c_1	0	0	1
c_2	0	0	1
c_3	0	1	1
c_4	1	1	1
c_5	0	0	1
c_6	1	0	0
c_7	0	0	0
c_8	1	0	0
c_9	0	0	0
c_{10}	0	0	0
c_{11}	1	0	0
c_{12}	0	0	0
c_{13}	1	0	0
c_{14}	1	0	0
c_{15}	0	0	0
c_{16}	1	0	0
c_{17}	0	0	0
c_{18}	0	1	1
c_{19}	1	1	1
c_{20}	1	1	1

Table 6. Hybrid Prototyping Selection Matrix

Criteria	v_{i1}	v_{i2}	v_{i3}
c_1	1	1	1
c_2	1	1	1
c_3	1	1	1
c_4	0	1	1
c_5	1	1	1
c_6	1	1	1
c_7	0	0	0
c_8	1	1	0
c_9	0	0	0
c_{10}	0	0	0
c_{11}	1	1	1
c_{12}	0	0	0
c_{13}	1	1	1
c_{14}	0	1	0
c_{15}	0	0	0
c_{16}	0	1	0
c_{17}	0	0	0
c_{18}	0	0	1
c_{19}	1	1	0
c_{20}	0	1	1

Table 7. Operational Specification Selection Matrix

Criteria	v_{i1}	v_{i2}	v_{i3}
c_1	0	1	1
c_2	0	1	1
c_3	0	1	1
c_4	0	1	1
c_5	0	1	1
c_6	1	1	0
c_7	0	0	0
c_8	1	1	0
c_9	0	0	0
c_{10}	0	0	0
c_{11}	1	0	0
c_{12}	0	0	0
c_{13}	1	1	0
c_{14}	0	1	0
c_{15}	0	0	0
c_{16}	0	1	0
c_{17}	0	0	0
c_{18}	0	0	1
c_{19}	1	0	0
c_{20}	0	1	1

Table 8. Characteristics of Example Project

Criteria	v_{i1}	v_{i2}	v_{i3}
c_1	*1*	*0*	*0*
c_2	*1*	*0*	*0*
c_3	*1*	*0*	*0*
c_4	*0*	*0*	*1*
c_5	*1*	*0*	*0*
c_6	*0*	*0*	*1*
c_7	*0*	*0*	*1*
c_8	*0*	*0*	*1*
c_9	*0*	*1*	*0*
c_{10}	*1*	*0*	*0*
c_{11}	*0*	*1*	*0*
c_{12}	*0*	*0*	*1*
c_{13}	*0*	*0*	*1*
c_{14}	*1*	*0*	*0*
c_{15}	*0*	*1*	*0*
c_{16}	*1*	*0*	*0*
c_{17}	*0*	*1*	*0*
c_{18}	*0*	*0*	*1*
c_{19}	*1*	*0*	*0*
c_{20}	*1*	*0*	*0*

Table 9. Calculation of Rating for the Conventional Process Model

Criteria	$(s_{i1} * a_{i1})$	$(s_{i2} * a_{i2})$	$(s_{i3} * a_{i3})$		
c_1	0	0	0	=>	0
c_2	0	0	0	=>	0
c_3	0	0	0	=>	0
c_4	0	0	1	=>	1
c_5	0	0	0	=>	0
c_6	0	0	0	=>	0
c_7	0	0	0	=>	0
c_8	0	0	0	=>	0
c_9	0	0	0	=>	0
c_{10}	1	0	0	=>	1
c_{11}	0	0	0	=>	0
c_{12}	0	0	1	=>	1
c_{13}	0	0	0	=>	0
c_{14}	1	0	0	=>	1
c_{15}	0	0	0	=>	0
c_{16}	1	0	0	=>	1
c_{17}	0	0	0	=>	0
c_{18}	0	0	1	=>	1
c_{19}	1	0	0	=>	1
c_{20}	1	0	0	=>	1
					8

Table 10. Process Model Ratings for Example Project

Process Model	Rating
Level 3	
Conventional	8
Incremental	8
Evolutionary	16
Level 2	
Waterfall	6
Hybrid Prototyping	10
Operational Specification	3

Step 3: Select the process model with the highest rating. For the level 3 process models the ratings indicate that the evolutionary process model would be the most appropriate choice for this project. For the level 2 process models, the hybrid prototyping process model is the best selection at this level of abstraction.

6 Conclusions

The selection criteria determined in this work are by no means definitive. Their definition resulted only from literature research and informal observation of many software development efforts. Valid correlations between project characteristics and alternative process models can only be done after performing correlation analyses among the characteristics, the use of various process models, and successes and failures for a large number of projects.

Further study is needed into the actual meaning of the score for a process model. There may be some benefit in calculating "negative" scores indicating the penalty for ignoring certain criteria or values. This could be useful for breaking ties or for evaluating the cost of not using the recommended process model. Criteria with significant "negative" influence also could prompt plans to avoid problems when the recommended process model is not used or to tailor a process model to fit all of the evaluations.

References

[1] Agresti, W., "What Are the New Paradigms?," Tutorial: New Paradigms for Software Development, W. Agresti, ed., Washington, D.C.: IEEE Computer Society Press, 1986, pp. 6-10.

[2] Alexander, L., *Selection Criteria for Alternate Software Life Cycle Process Models*, Software Systems Engineering M.S. Thesis, Fairfax, Virginia: George Mason University, 1990.

[3] Balzer, R., T. Cheatham, Jr., and C. Green, "Software Technology in the 1990's: Using a New Paradigm," *IEEE Computer*, 6, 11 (November 1983), pp. 39-45.

[4] Basili, V. and H. Rombach, "Tailoring the Software Process to Project Goals and Environments," *Proceedings Ninth International Conference on Software Engineering*, Washington, D.C.: IEEE Computer Society Press, 1987.

[5] Bersoff, E., B. Gregor, and A. Davis, "A New Look at the C^3I Software Life Cycle," *SIGNAL*, 41, 8 (April 1987), pp. 85-93.

[6] Boehm, B., "Software Engineering," *IEEE Transactions on Computers*, 25, 12 (December 1976), pp. 1226-1241.

[7] Boehm, B., "A Spiral Model Of Software Development and Enhancement," *IEEE Computer*, 21, 5 (May 1988), pp. 61-72.

[8] Davis A., E. Bersoff, and E. Comer, "A Strategy for Comparing Alternative Software Development Life Cycle Models," *IEEE Transactions on Software Engineering*, 14, 10 (October 1988).

[9] Davis Alan M., Software Requirements: Analysis and Specification, Englewood Cliffs, New Jersey, Prentice Hall, 1990.

[10] Davis, A., "Operational Prototyping: The POST Story," submitted to *IEEE Software*, January 1991.

[11] Fairley, R., Software Engineering Concepts, New York, New York: McGraw-Hill, Inc., 1985.

[12] Gilb, T., "Evolutionary Delivery versus the Waterfall Model," *ACM Software Engineering Notes*, 10, 3 (July 1985), pp. 49-61.

[13] Gomaa, H., "Impact of Rapid Prototyping on Specifying User Requirements," *ACM SIGSOFT Software Engineering Notes*, 8, 2 (April 1983).

[14] Luqi, "Software Evolution Through Rapid Prototyping," *Computer*, 22, 5 (May 1989).

[15] Royce, W., "Managing the Development of Large Software Systems: Concepts and Techniques," *Proceedings, WESCON*, August, 1970, reprinted in: *Proceedings Ninth International Conference on Software Engineering*, Washington, D.C.: IEEE Computer Society Press, 1987, pp. 328-338.

[16] Sitaram, P., *Concurrent Modeling of Software*, Software Systems Engineering M.S. Thesis, Fairfax, Virginia: George Mason University, 1990.