Chapter 6

Model: POM3

This chapter describes POM3, a model used to identify problems of Stochastic methods and also to benchmark CrossTree. This chapter starts with history of POM models and explains the motivation of designing better versions of initial model. Chapter concludes with confirming problems identified in last chapter in applying stochastic methods.

6.1 POM3

POM3 is a software estimation model like XOMO for Software Engineering. It is based on Turner and Boehm's model of agile development. [6, 7]. It compares traditional plan-based approaches to agile-based approaches in requirements prioritization. It describes how a team decides which requirements to implement next. POM3 reveals requirements incrementally in random order, with which developers plan their work assignments. These assignments are further adjusted based on current cost and priority of requirement. POM3 is a realistic model which takes more runtime than standard lab models(2-100ms, not 0.006-0.3ms) [44].

6.2 History

Port, Oklov, Menzies (POM) simulated a fundamental, yet intangible benefits of requirements prioritization strategies. The simulation was based on detailed empirical studies which was later improvised in POM2 and POM3.

6.2.1 POM1

Initial model of POM was a partial implementation of Boehm, and Turner which explored the effects of different prioritization policies while adjusting rate at which new requirements arrive/change. Noor, Rabiser and Grunbacker [65] explored trade-offs in planning agile processes and also criticized POM to explore only one of five scales proposed by Boehm and Turner. POM2 was a result of answering the problems of POM1 to address all 5 scales of Boehm. POM1 assumed small projects with 10 developers per project and maximum of 25 requirements.

6.2.2 POM2

POM2 implements all five scales compared to POM1 which just implements one scale proposed by Boehm and Turner. POM1 implemented one team whereas POM2 implements multiple teams. POM2 has dependents of requirements where child requirements must be satisfied before completing parents requirements. POM2 simulates projects with up to 300 developers and expanded number of requirements to 750.

The following are the inputs that effect the processing within POM2 mentioned in [46]:

- In high critical systems, requirements cost more to develop.
- In high dynamism, the requirements cost and values change frequently.

6.3 POM3 attributes

The catch of POM3 is its highly dynamic so the cost and priority can change after developers have made their work assignments. So POM3 is simulated such that some developers spend some time focusing on wrong requirements as well. It is considered pragmatically that in highly dynamic environments, the agile is better method for reacting to arrival of new requirements and existing requirements changing value. Most of the literature debates agile vs plan based models which can lead to false preconception of practices using interesting combinations of both. A more nuanced view is offered by Boehm and Turner. They define five scales that can characterize the difference between plan-based and agile-based methods:

- 1. Project size
- 2. Project Criticality (higher criticality higher the cost)
- 3. Requirements dynamism (priorities and costs change)
- 4. Personnel (developer skill level)
- 5. Organizational culture (larger the number more conservative the team is)

6.4 POM3

Requirements are represented as a set of trees in POM3. Single node of the tree represents a single requirement. These nodes are combined into trees which actually form the requirements heap. Each node i.e. requirement consists of the following:

- 1. Prioritization value
- 2. Prioritization cost
- 3. Child requirements

- 4. Dependencies

The prioritization values are in the range of 30 to 500 and cost ranges from 1 to 100. These values are chosen in consultation with Richard Turner which were the same as used in [44]. Some percent of requirements are marked as visible initially leaving the rest to be revealed by POM3 as the project progresses.

In POM3, teams divide the tasks of completing project's requirements. The number of tasks assigned to a team is relative to the size of the team. The team size is an decision input and remains constant throughout the simulation. POM3's personnel are divided into three categories based on the programming skill: 1.Alpha 2.Beta 3.Gamma. Alpha programmers are generally the best, most-paid type of programmers while Gamma programmers are the least experienced, least-paid. The ratio of personnel type follows the personnel decision as set out by the Boehm and Turner [6,7] in following table:

	project size				
	0	1	2	3	4
Alpha	45%	50%	55%	60%	65%
Beta	40%	30%	20%	10%	0%
Gamma	15%	20%	25%	30%	35%

Costs are updated after teams are generated and assigned to requirements. Criticality and Criticality modifier play a major role in deciding those cost changes. Criticality affects the cost-affecting nature of the project and ensures how safety-critical a project is. Whereas, criticality modifier indicates a percentage of teams affected by safety critical requirements. In the formula, Cm is a criticality modifier:

$$cost = cost * {C_M}^{criticality} \tag{6.1}$$

After generating teams and requirements for those teams to work on, POM3 runs through a shuffling process. This shuffling process is repeated at random for ($1 \leq N \leq 6$) times, where N is chosen at random. The shuffling process includes following steps:

- *Collect Available Requirements*: The assigned requirements are searched for all available requirements that can be completed by each individual team. These requirements must be available to complete i.e. without any unsatisfied child requirements or unsatisfied dependencies. The team budget is then updated by calculating the total cost of requirements remaining for the team and dividing by the number of shuffling iterations:

$$team.budget = team.budget + totalCost/numShuffles$$

- *Apply for a Requirements Prioritization Strategy*: The available requirements with high priority should be satisfied first. In order to support that, POM3 sorts the available requirements by some sorting strategy which includes requirement's cost and values. A plan decision determines this strategy and is implemented inside POM3.

- *Execute Available Requirements*: The team executes the available requirements that are sorted in previous step sequentially. Some requirements may not be executed due to the budget allocations which are executed in the next iteration.

- *Discover New Requirements*: Upon completing few requirements, there is a possibility to open new requirements that depended on these completed requirements. To model the probability of arrival of new requirements, input decision Dynamism is considered to be used in a Poisson distribution. The following formula defines the percentage of new requirements to be added in the heap:

$$new = Poisson\left(dynamism/10\right) \tag{6.2}$$

Short name	Decision	Description	Controllable
Cult	Culture	Number (%) of requirements that change.	yes
Crit	Criticality	Requirements cost effect for safety critical systems (see Equation 6.1 in the appendix).	yes
Crit.Mod	Criticality Modifier	Number of (%) teams affected by criticality (see Equation 6.1 in the appendix).	yes
Init. Kn	Initial Known	Number of (%) initially known requirements.	no
Inter-D	Inter-Dependency	Number of (%) requirements that have interdependencies. Note that dependencies are requirements within the *same* tree (of requirements), but interdependencies are requirements that live in *different* trees.	no
Dyna	Dynamism	Rate of how often new requirements are made (see Equation 6.2 in the appendix).	yes
Size	Size	Number of base requirements in the project.	no
Plan	Plan	Prioritization Strategy (of requirements): one of 0= Cost Ascending; 1= Cost Descending; 2= Value Ascending; 3= Value Descending; 4 = $\frac{Cost}{Value}$ Ascending.	yes
T.Size	Team Size	Number of personnel in each team	yes

Figure 6.1: List of Decisions used in POM3. The optimization task is to find settings for the controllables in the last column. Taken from [44]

- *Adjusting Priorities*: The teams adjust their requirements priorities by making use of input decisions Culture and Dynamism. The following formula helps adjust normal distribution and is scaled by project's Culture:

$$value = value + maxRequirementValue * Normal(0,D) * C$$

Finally, POM3 models score a planning policy by comparing a developer's incremental decisions to those that might have been made by some omniscient developer. This omniscient developer has access to final cost and priority of requirements.

6.5 POM3 submodels

Several submodels of POM3 are defined to give different modes of POM3. In summary, POM3a is an unconstrained version of POM3 with its full decision ranges for all decisions. POM3a represents both classes of large and small projects. POM3b is constrained with decisions for only small projects with high criticality and POM3c is designed for Large projects with high dynamism. There is no claim that this represents all possible kinds of projects. Rather, these three sub-models mark as interesting variety of projects to study. Figure 6.2 defines decisions of these submodels [44] defined below in detail.

	POM3a A broad space of projects.	POM3b Highly critical small projects	POM3c Highly dynamic large projects
Culture	$0.10 \leq x \leq 0.90$	$0.10 \leq x \leq 0.90$	$0.50 \leq x \leq 0.90$
Criticality	$0.82 \leq x \leq 1.26$	$0.82 \leq x \leq 1.26$	$0.82 \leq x \leq 1.26$
Criticality Modifier	$0.02 \leq x \leq 0.10$	$0.80 \leq x \leq 0.95$	$0.02 \leq x \leq 0.08$
Initial Known	$0.40 \leq x \leq 0.70$	$0.40 \leq x \leq 0.70$	$0.20 \leq x \leq 0.50$
Inter-Dependency	$0.0 \leq x \leq 1.0$	$0.0 \leq x \leq 1.0$	$0.0 \leq x \leq 50.0$
Dynamism	$1.0 \leq x \leq 50.0$	$1.0 \leq x \leq 50.0$	$40.0 \leq x \leq 50.0$
Size	x ∈ [3,10,30,100,300]	x ∈ [3, 10, 30]	x ∈ [30, 100, 300]
Team Size	$1.0 \leq x \leq 44.0$	$1.0 \leq x \leq 44.0$	$20.0 \leq x \leq 44.0$
Plan	$0 \leq x \leq 4$	$0 \leq x \leq 4$	$0 \leq x \leq 4$

Figure 6.2: Three classes of projects studied using POM3. Taken from [44]

- POM3a: The unconstrained version with full ranges of all decisions
- POM3b: Small Projects, High criticality.
- POM3c: Large Projects, High Dynamism.

6.6 Implementing NSGA-II with POM3

The parameters used by NSGA-II are same as ones used by Krall [44] as explained in section 2.3.3. POM3 model is built on nine attributes which act as nine dimensions to predict three objectives. Similar to XOMO parameters explained in section 5.3.1 the only parameter modified in this thesis is the number of generations. A default of 500 is maintained if number of generations are not mentioned (i.e $n = 500$). The objectives being:

- Cost: Cost of project
- Completion: Completion time of project
- Idle: Idle time of project.

Similar, to standard search algorithms, NSGA-II on POM3 starts with generating an initial population N from POM3. This solution space is explored by NSGA-II by first sorting them by elite