Applications of Simulation and AI Search

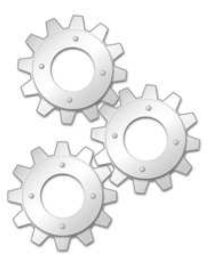
Bryan Lemon, Aaron Riesbeck, Tim Menzies, Justin Price, Joseph D'Alessandro, Rikard Carlsson, Tomi Prifiti, Fayola Peters, Hiuhua Lu, Dan Port*
Lane Department of Electrical Engineering and Computer Science, West Virginia University, Morgantown, WV 26506-6109, USA
*Information Technology Management, University of Hawaii, Honolulu, HI 96822-2457, USA

SUMMARY

Why is so much time wasted on evidence-less debates on software process when a simple combination of simulation plus automatic search can mature the dialogue much faster?

Here, we implemented Boehm-Turner's model of agile and plan-based software development. That tool is augmented with an AI search engine to find the key factors that predict for the success of agile or traditional plan-based software developments. According to our simulations and AI search engine: (1) in no case did agile methods perform worse than plan-based approaches; (2) in most cases, agile performed much better. Hence, we recommend that the default development practice for organizations be an agile method.

SIMULATOR STRUCTURE



POM2 shuffles requirements through three stages: "to-do", "planned", and "completed". *Requirements* are represented as cost/value pairs. *Projects* are represented as requirements organized into acyclic trees. Requirement dependencies are represented by edges between nodes, with the constraint that all children must be completed before parents. Requirement nodes are assigned children based on an exponential distribution: $\frac{1}{X}\%$ of nodes have X children.

A *team* contains a varying number of members. Each team is assigned a subtree from within the project tree, and maintains a *plan* which represents the set of requirements to be completed next iteration. The plan's size is determined by the *team's budget*, and requirements in the plan are selected from team's project subtree. Work is completed in *iterations*. A team completes a certain number of requirements per iteration. During each iteration, it is possible for a team to "discover" new requirements. The number of new requirements are calculated using Eq. 1:

$$size * 2.5 \tag{1}$$

Inter-team cooperation is modelled by creating dependencies between trees at the same level. All requirements in a team's plan are ordered by a *prioritization policy*. A project is completed when all subtrees are completed or management orders early termination

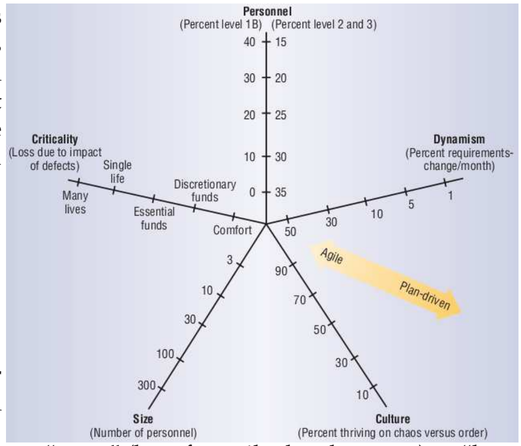
PRIORITIZATION POLICIES

A prioritization policy determines how many requirements are initially known and how the plan is modified when new requirements arrive.

POM 2 explores two prioritization policies:

1. Plan Based (PB) : A non-agile development method where requirements are sorted once at the start of development using the original $\frac{value}{cost}$ numbers assumed at the start of the project.
2. Agile2 (AG2) : A development method where requirements are sorted every iteration on policies sort requirements (every iteration) on $\frac{value}{cost}$.

DEFINING THE MODEL



POM2 implements four of the five scales identified by Boehm and Turner that distinguish agile from traditional plan-based projects:

- Criticality
- Dynamism
- Culture
- Size

Boehm and Turner measure criticality in terms of losses due to defects, it ranges from "none" (best for agile development) to "loss of many lives" (must be carefully planned, suited for plan-based approach). In POM2 X% of the requirements are affected by criticality, where $X = 2...10$. Each requirement in a team's tree is adjusted using the equation:

$$cost' = cost * X^{criticality} \tag{2}$$

Dynamism measures how frequently new requirements are created and their values change. Boehm and Turner measure dynamism in terms of percent of requirements changed each month, and has the range 50% (best for agile) to 1 (best for plan-based). In POM2 we implement this as follows:

- Initially only 30% to 70% of the requirements in the project tree are "visible".
- After each iteration we make visible $new = Poisson(\lambda)$ more requirements.
- Dynamism also affects the value of already visible requirements. The value is adjusted using the equation:

$$value' += maxValue * N(0, \sigma) \tag{3}$$

- After setting σ , we set λ to 10% of σ.

Culture, according to Boehm and Turner is measured in terms of the percent of the staff thriving on chaos, ranging from 10% (best for plan-based) to 90% (best for agile). In POM2 culture is handled through an accepted value associated with each requirement. The calculation for the accepted value is:

$$value + (valueN(0, \sigma) * culture) \tag{4}$$

Size is measured in terms of the number of personnel and has a range of 3 (best for agile), 10, 30, 100, 300 (best for plan-based). In POM2 this size is picked randomly and dictates the number of requirements in the project:

$$number\ of\ requirements = size * 2.5 \tag{5}$$

The one scale we did not implement was personnel. Boehm and Turner describe personnel using the Cockburn mixtures model of Alpha, Beta and Gamma programmers:

- Alpha: Most productive/flexible programmers.
- Beta: Able to perform discretionary method steps.
- Gamma: Unable/unwilling to follow shared methods.

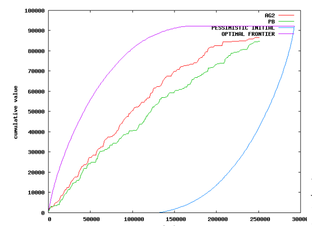
By the conventions of the Boehm and Turner personnel scale, lower personnel values indicate more alpha programmers. Combining the COCOMO effort multipliers and the Cockburn mixtures model, the net effect is nearly zero. For that reason, POM2 ignores personnel and will address it in future work.

SIMULATOR SCORING

POM2 defines three different $cost/value$ curves of the completed requirements when scoring the project:

1. Optimal Frontier(OF): The completed requirements are ordered by their final $cost/value$ in the optimal descending order.
2. Pessimistic Frontier(PF): The completed requirements are ordered by their final $cost/value$ in ascending order.
3. Actual Curve(AC): The completed requirements are ordered by the order they were completed in the simulator.

The score is then defined as the area between the OF and PF (Eq. 6).



$$\texttt{score} = \frac{\int AC - \int PF}{\int OF - \int PF} \tag{6}$$

By using the area between the OF and PF curve, the project configuration bears no impact on the final score. Thus leaving just the impact of the model inputs and prioritization policy.

MODEL INPUTS

POM2 accepted eight input variables. *Criticality*: affects requirements development cost. *Criticality Modifier*: affects the severity of criticality. *Culture*: affects requirements reordering and perceived value. *Initial-known*: sets the size of the initial requirements set. *Inter-dependency*: frequency of inter-team dependency for requirements. *Size*: size of project (number of requirements). *Team size*: size of teams working on project. *Dynamism*: how frequently requirements' values change.

RESULTS

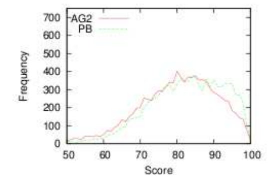
policy / dynamism		bin 1	bin 2	bin 3	bin 4	bin 5	bin 6	bin 7	bin 8	bin 9	bin 10
	criticality	.82	.86	.91	.95	1.0	1.04	1.08	1.13	1.17	1.22
	criticality modifier	2.0	2.8	3.6	4.4	5.2	6.0	6.8	7.6	8.4	9.2
	culture	0	10	20	30	40	50	60	70	80	90
	initial known	.40	.43	.46	.49	.52	.55	.58	.61	.64	.67
	inter-dependency	0	10	20	30	40	50	60	70	80	90
	size (total personnel)	3.0	32.7	62.4	92.1	121.8	151.5	181.2	210.9	240.6	270.3
	team size (people per team)	1.0	5.1	9.2	13.3	17.4	21.5	25.6	29.7	33.8	37.9
plan-based / very low	criticality										
$\sigma = \lambda = 0$	criticality modifier										
	culture										
	initial known									2	4
	inter-dependency							1	1	1	
	size	89	2								
	team size										1
plan-based / medium	criticality								5	21	52
$\sigma = 0.15, \lambda = 0.0.15$	criticality modifier										
	culture										
	initial known										
	inter-dependency						1	2	6	6	8
	size										
	team size										1
plan-based / very-high	criticality							1	6	22	58
$\sigma = 2, \lambda = 0.2$	criticality modifier										
	culture										
	initial known										
	inter-dependency						2	3	6	10	11
	size										
	team size										1
agile 2 / very low	criticality										
$\sigma = \lambda = 0$	criticality modifier										
	culture										
	initial known							1	4	12	27
	interdependency						1	4	6	5	3
	size	72	1								
	team size									1	4
agile 2 / medium	criticality	1						1	1	1	1
$\sigma = 0.15, \lambda = 0.015$	criticality modifier							1	1	1	1
	culture						3	10	19	22	29
	initial known					1		2	2	2	2
	interdependency				1	1	1	1		1	
	size	97									
	team size		17	20	11	6	1				
agile 2 / very high	criticality	1				1		1	1	1	1
$\sigma = 2, \lambda = 0.2$	criticality modifier	1	1	1	1	1	1	1		1	1
	culture					1	4	13	17	20	22
	initial known						1	1	1	1	1
	interdependency		1	1	1	1	1	1	1		1
	size	100									
	team size		15	18	11	5	2	1			

HOW WE SEARCH THE MODEL

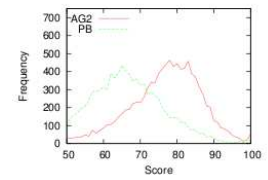
POM2 presents a 10 dimensional hypothesis space. To find the critical factors, POM2 uses the KEYS range subset selector. KEYS uses Bayesian contrast set learning to find the single range that most selects for the preferred outcomes. Subsequent iterations of KEYS constrains the simulator to always include that range. The smallest subset of selected ranges that most improve the final output is the *model treatment*.

WHAT WE FOUND

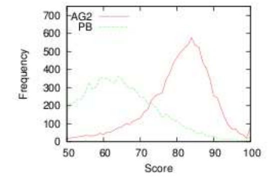
Low dynamism: $\sigma = 0, \lambda = 0$



Medium dynamism: $\sigma = 0.15, \lambda = 0.015$



High dynamism: $\sigma = 2, \lambda = 0.2$



We ran KEYS 1000 times for each combination of very low, medium, very high dynamism and plan-based, agile 2 prioritization policy. The findings are summarized in §RESULTS. Several majority case effects deserve note:

- Criticality Modifier: Never chosen in more than 1% of the cases.
- Initial Known/Inter-Dependency: Never chosen in > 50% of the runs.
- Team Size: Never < 5 team members.
- For Agile 2:
  - Size: The smallest size was chosen in 72% – 100% of the runs.
  - Culture: The upper bins were chosen in 72% – 80% of the runs where dynamism was a factor.
  - Team Size: Should be 5 - 17 for medium and very high dynamism.

The 1000 runs of KEYS shows that many results are very similar (e.g. all the Agile 2 results offer nearly the same pattern). KEYS shows that there are two major divisions of its 10-dimensional space: very high criticality and very small team size. We ran 10,000 simulations in the union of these two divisions (shown above). The median score of Agile 2 is shows stability under changing dynamism, while the median score of Plan-Based falls as dynamism increases. We see that the median score of Agile 2 is equal to or greater than the median score of Plan-Based development.

We conclude that in the general case, Agile 2 out performs Plan-Based development. Only rarely does Plan-Based out perform Agile 2.

RECOMMENDATIONS

1. The default development practice should be an Agile 2 method.
2. Debates about software process can be greatly shortened, and improved, by combining process simulation and AI search.

FOR MORE INFORMATION

Bryan Lemon (bryan@bryanlemon.com)
Tim Menzies Ph.D. (tim@menzies.us)
Joseph D'Alessandro (jdalessa57@gmail.com)

SUPPORT

This research was carried out at West Virginia University under a grant from the National Science Foundation (NSF CISE 71608561).