Quantifying Risk In Waterfall Methodology: Case Study At Aplikasi Super

Mengukur Risiko Dalam Metodologi Waterfall: Studi Kasus Di Aplikasi Super

Indriati Njoto Bisono[1], Vincent Arvin[2], Hanijanto Soewandi[3]
Teknik Industri, Fakultas Teknologi Industri, Universitas Kristen Petra, Surabaya[1]
Aplikasi Super, Nusantara Technology, Surabaya[2]
MicroStrategy, Tysons Corner, VA, USA[3]
mlindri@ petra.ac.id[1], vincent.arvin@nusantara.technology[2], hsoewandi@microstrategy.com[3]

**Corresponding Author*

ABSTRACT
In this paper, we presented a very simple mathematical model based on Dynamic Programming to answer one of the biggest concern in Waterfall methodology, namely: quantifying risk. Our approach essentially resembles Elmaghraby (2005), but we have more stages and use uniform distributions. With this approach, we can show and quantify the risk in Waterfall methodology so that decision maker can understand the implication of his decision. Dynamic Programming solution also provides a blue print for adjusting decision if the early (previous) stage does not go as planned. A case study at Aplikasi Super with some sensitivity analysis is provided as a numerical illustration.
Keywords: *Waterfall Methodology, Risk Management, Sequential Decision Process, Dynamic Programming*

ABSTRAK
Dalam artikel ini, kami menyajikan model matematis yang sederhana berdasarkan Dynamic Programming untuk menjawab salah satu kekhawatiran terbesar dalam metodologi Waterfall, yaitu: mengukur risiko. Pendekatan kami pada dasarnya mirip dengan Elmaghraby (2005), tetapi dengan lebih banyak tahapan dan menggunakan distribusi uniform. Dengan pendekatan ini, kami dapat menunjukkan dan mengukur risiko dalam metodologi Waterfall sehingga pengambil keputusan dapat memahami implikasi dari keputusan mereka. Solusi Dynamic Programming juga menyediakan cetak biru untuk menyesuaikan keputusan jika tahap awal (sebelumnya) tidak berjalan sesuai rencana. Studi kasus di Aplikasi Super dengan beberapa analisis sensitivitas disertakan sebagai ilustrasi numerik.
Kata Kunci: Metodologi *Waterfall*, Manajemen Risiko, Proses Keputusan Berurutan, Pemrograman Dinamis

1. Introduction

Aplikasi Super, part of PT Krakatau Karya Abadi, is an application-based startup business that primarily provides a wide basic-need chain to provide equal distribution for the second and third-tier cities in Indonesia. As the first social commerce platform in Indonesia that is ISO 9001:2015 certified, Aplikasi Super aims to solve economic inequality across cities for Indonesia’s future economy. Aplikasi Super is also the first consumer technology company in Indonesia backed by Y-Combinator, which oversees the main feature, SuperAgen, which is agent-led commerce that enables community leaders to become retailers within the communities. One of us was lucky enough to be given a handful of work as an Associate Product Manager in the Product and Technology Department. The department is responsible for planning, conceptualizing, and executing requests from other departments, including designing the logical framework to satisfy the requests from the corresponding departments as well as getting approval from the other related departments and stakeholders.

Generally speaking, software developments will always exist, especially in technology companies such as: Aplikasi Super. There will always be a need for improvised applications and web dashboards at all times. At Aplikasi Super, the web dashboard needs to be improved as the business grows. There are several problems which might need a web dashboard improvement and feature creations. For example: an inefficient delivery system, complicated manual product bundle creation, and the need to have a discount feature. With an improved dashboard, those inefficiencies might be overcome, or at least mitigated. In terms of incompleteness, there are always some features that the users need, but are not available yet. Furthermore, there are some procedures that still need to be done manually. These problems lead to the usage of a dashboard and its features as the main platform for Aplikasi Super to execute its business processes.

In improving the dashboard to support better business processes, the requests are given by other departments. Hence, those requests generally came from the head of other departments (on business side) – where requests are mainly from each individual's ideas and thoughts. Even though, detailed data regarding the reason for requests were not available, the requests are usually well understood (or can be easily verified). Furthermore, given that other departments are also very busy, enough freedom & trust is given to develop the improvement. Given all the above constraints and flexibility given, Waterfall methodology in project management seems to be a more appropriate approach. Hence, the choice.

Our biggest problem is how to manage the risk associated with this Waterfall methodology (see Literature Review for detail). When we ask our individual team member about estimate of a task i, the typical answer that we got is: "I can finish it between $lb_{i,1}$ and $ub_{i,1}$ days." Furthermore, when we ask how about if they (q people) work together for activity i, we got the answer such as: "Oh, we can then finish it between $lb_{i,2}$ and $ub_{i,2}$ days." The interesting aspect is: we find out that $lb_{i,2} \neq \frac{lb_{i,1}}{n_i}$ and $ub_{i,2} \neq \frac{ub_{i,1}}{n_i}$. But, rather $lb_{i,2} = \frac{k_i \times lb_{i,1}}{n_i}$ and $ub_{i,2} = \frac{k_i \times ub_{i,1}}{n_i}$ with $k_i > 1$. We will have more discussion about this in the mathematical modeling section.

In this paper, we demonstrate a very simple yet realistic mathematical & computational model using Dynamic Programming (DP) as an answer to the problem, and we explain how Waterfall methodology with proper risk management is used at Aplikasi Super to create additional functionalities (enhance) in its web dashboard as a platform to execute most business processes.

2. Literature Review

In many books, the typical diagram of the Waterfall model was attributed to Royce (1970). But, many years before, Torres & Benington (1956) gave a presentation at the Symposium on Advanced Programming Methods for Digital Computers with a similar top-down concept. It should be noted that all authors, q failure" while Torres & Bennington pointed out that the process was not in fact performed in a strict top-down fashion, but depended on a prototype. Bell & Thayer (1976) is perhaps the first that refer the qq as "waterfall". The general picture of Waterfall methodology is given in Figure 1.

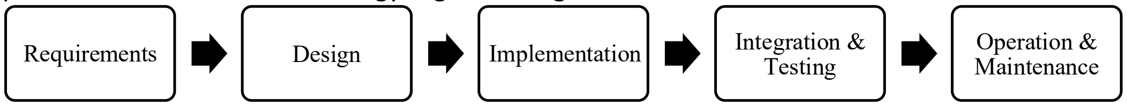

Figure 1. Waterfall Methodology

Eventhough, Waterfall methodology had been around for a while, surprisingly very little literature were found on the project planning aspect of it. Most literature review that we can find usually contrast between Waterfall *vs.* Agile methodology, *e.g.*, van Casteren (2017), Thesing *et.al.* (2021), & Chandran & Das Aundhe (2022). No one seems to response to the flaws that even Royce as well as Torres & Bennington had raised earlier, *i.e.*, how to measure and mitigate the risk associated with testing at the very last stage.

Very likely, these are caused by the fact that the project planning for Waterfall methodology seems to be very straightforward with a single path with m activities (in Figure 1, clearly $m = 4$ (we assume Operation & Maintenance is a whole new project within software development life cycle).

In the context of Project Management, Elmaghraby (2005) was among the first who gave a warning about the fallacy of average when it comes to project risk management. Our work is heavily influenced by his. But, given that we don't know the distribution of each activity, and we are just told about lower bound lb_i and upper bound ub_i for activity i, we will assume a uniform random variable rather than exponential random variable as in Elmaghraby (2005). This is one of the main difference between his & ours. We will not review Dynamic Programming since this method has been widely covered in many introductions to Operations Research books, *e.g.*, Hillier & Lieberman (2021). However, it is important to point out that clever implementation of DP such as in Knapsack problem and/or Wagner-Whitin algorithm for Economic Lot Scheduling Problem iterates on the knapsack capacity (or time). We follow the Elmaghraby approach iterating over the starting time of each activity in this paper.

The convolution of multiple uniform random variables has been heavily studied for many years. In fact, the sum of n uniform distribution $U(0,1)$ is known as Irwin-Hall distribution, named after Joseph Oscar Irwin and Philip Hall, who wrote about it independently in 1927 (see Irwin (1927) and Hall (1927) for detail). Irwin-Hall distribution has its own usage, but for our purpose, we deal with activities duration (which is a random variable) that is distributed uniformly between lb_1 and ub_1, *i.e.*, $U(lb_1, ub_1)$.

For n uniform distribution where each of them is uniformly distributed according to $U(lb_i, ub_i)\ \forall i = 1, \dots, n$, we also have papers such as: Olds (1952), Ueda *et.al.* (1994), Killmann & von Collani (2001), and Bradley & Gupta (2002). We point out Olds (1952) or Killmann & von Collani (2001, Theorem 2.2) that can be used for our purpose. But, since we plan to do calculations differently, we do not need to elaborate further here.

3. Data and Model

Modeling for Work Content

It should be very easy and obvious to see that, if we assume that we declare a software as feature complete/generally available (GA) after satisfactory integration testing, then the Waterfall methodology in Figure 1 can be represented as the summation of 4 uniform random variables, *i.e.*, W_1, W_2, W_3, and W_4, where: W_1 is the work content for requirement gathering (which is normally done by Product Manager), W_2 is the work content for architectural software design (which is normally done by Software Architect), W_3 is the work content for the implementation (coding) that is normally done by Software Engineer (SE), and W_4 is the work content for integration and testing (that is normally done by Software Quality Engineer or Software Engineer in Test). Upon completion of the work context W_4, we can declare the feature complete (or declare the product generally available). With this, we essentially have: $W_i \sim U(lb_{i,1}, ub_{i,1})\ \forall i = 1, 2, 3, \& \ 4$. The subscribe $(i, 1)$ indicate that the value of lower bound (and upper bound) for activity i if it is done by 1 person.

In our teams at Aplikasi Super, we work together in a group with the following numbers of people from each stage (see column m_i in Table 2) and we are asked to complete the feature in $T_{max} = 20$ days (due date is about 1 month, *i.e.*, 4 working weeks). Various information about lower bound lb_i and upper bound ub_i for each activity i are obtained based on a question and answer like we have explained before. One of us works as an Associate Product Manager during the process and completes multiple feature releases using this Waterfall methodology over six-month period. The unit cost for each activity i, *i.e.*, c_i is normalized to maintain privacy and confidentiality. As we have briefly explain in the beginning, we found out that when we add resources to an activity i, its lower bound and upper bound reduce. For simplicity, we assume and model the work content as:

$$W_{i,j+1} \sim n_{i,j} p_{i,j\rightarrow j+1} U_{i,j} \ \forall i = 1,2,3,4 \ and \ \forall j = 1,2,\dots,m_i$$

where: $p_{i,j\rightarrow j+1}$ is a *people* (multiplication) factor when we increase resource to work on activity i by 1.

Table 1. Resources, their Unit Costs, & Work content in one of Aplikasi Super feature

		$m_i=1$		$m_i=2$		$m_i=3$		$m_i=4$			
Activity *i*	m_i	$lb_{(i,1)}$	$ub_{(i,1)}$	$lb_{(i,2)}$	$ub_{(i,2)}$	$lb_{(i,3)}$	$ub_{(i,3)}$	$lb_{(i,4)}$	$ub_{(i,4)}$	$p_{i,j\rightarrow j+1}$	c_i
Requirement	$m_1=3$	4	8	3	6	2	5	NA	NA	$p_{1,j\rightarrow j+1} \approx 0.75$	1.7
Design	$m_2=2$	2	4	2	3	NA	NA	NA	NA	$p_{2,j\rightarrow j+1} \approx 0.80$	1.5
Implementation	$m_3=4$	10	15	7	11	5	8	4	6	$p_{3,j\rightarrow j+1} \approx 0.70$	1.4
Testing	$m_4=2$	5	10	4	8	NA	NA	NA	NA	$p_{4,j\rightarrow j+1} \approx 0.80$	1.2
	Completion Time =	21	37	16	28	11	24	12	22		

Data in Table 1 is used to illustrate our complete calculation as well as how we plan using Waterfall methodology in our feature development. To make it realistic, we actually rounded the value to the nearest integer after the multiplication. To further clarify, we have the following situations to illustrate:

- During the requirement gathering, if it is done by 1 Product Manager, the work content is given by $W_1 \sim U(4,8)$ man-days. But, it is done by 2 Product Managers, the work content becomes $W_1 \sim 2 \times U(3,6)$ man-days, and lastly if all 3 Product Managers collect the requirement, the work content further reduces to $W_1 \sim 3 \times U\left(2\frac{1}{4}, 4\frac{1}{2}\right) \approx U(2,5)$ days. First, notice that $lb_2 = 0.75 \times 4 = 3$ and $lb_3 = 0.75 \times 3 = 2\frac{1}{4} \approx 2$ (when there are 2 and 3 Product Managers working together respectively). Notice also that as we add more people to work, the expected duration of the activity becomes shorter: the mean duration reduces from $\frac{4+8}{2} = 6$ days (if it is done by 1 person) to become $\frac{3+6}{2} = 4.5$ days (if it is done by 2 persons) and it finally bests at $\frac{2+5}{2} = 3.5$ days if all 3 product managers work on it.
- Similarly, for the testing stage, if it is done by 1 Software Quality Engineer (SQE), the work content is given by $W_4 \sim U(5,10)$. However, when it is done by 2 SQEs, the work content becomes $W_2 \sim 2 \times U(4,8)$. Notice that in this case, the duration reduces from 7.5 days (if it is done by 1 person) to 6 days (when 2 persons are doing the testing).

It is worth mentioning that this modeling is very realistic as we reiterate the now (in)famous "Mythical Man Month" book by Fred Brooks (1975) that won the Turing award. Notice that the *people* (multiplication) factor $p_{i,j\rightarrow j+1}$ allows us to stretch or contract the Work content random variable to be different at different resources level – this is the essence of Brooks' law. For example: Consider an activity to design a feature. It is very possible that adding more people will cause the activity to be longer due to communication, *etc*. In fact, in the most general form, we can simply stretch and contract the work content random variable as we add 1 additional resource, *i.e.*, we have: $W_{i,j+1} \neq W_{i,j} \ \forall i = 1,2,3,4 \ and \ \forall j = 1,2,\dots,m_i$.

Modeling for Cost on Resources

Following Elmaghraby (2005) and given that requirement gathering, software design, implementation (coding), and integration (testing) are normally done by different people (with different salary), we define the cost to complete the feature to be:

$$TC_R = \sum_{i=1}^{4} c_i W_i = \sum_{i=1}^{4} c_i n_{i,j} U_{i,j} = c_1 n_{1,j} U_{1,j} + c_2 n_{2,j} U_{2,j} + c_3 n_{3,j} U_{3,j} + c_4 n_{4,j} U_{4,j}$$

where: c_i and $W_{i,j}$ are the unit cost (per person per day) and workcontent (man days) for activity i, and $W_i = n_{i,j}U_{i,j}$ – the work content is the random variable duration $U_{i,j}$ done by a $n_{i,j}$ person.

To illustrate this cost, consider an example with $c_1 = 1.7$, $c_2 = 1.5$, $c_3 = 1.4$, and $c_4 = 1.2$ respectively (as in Table 1). Please note that we do not need the actual cost (in IDR), we just need the relative value of one to the others.

Modeling for Cost of Tardiness

This cost is more difficult to measure in reality of software development. One possible approach (to quantify it) is to measure the opportunity loss if the feature development is not done on time. It will be easier if there is a binding contract with a clear penalty. Nonetheless, management can associate a cost number, *e.g.*, the unit time bonus of the entire group that works on this feature when it is delivered late by $(t - T_{max})$ days. For our purpose, we the cost of tardiness will be by the following form:

$$TC_{Tardiness} = c_T \times max\{0, t - T_{max}\}$$

where: c_T is the unit cost of tardiness (per day). For our case, the managerial decide to use the team bonus which is set to be $c_T = 5$ per day (again this is normalized & relative measure with respect to c_i.

Sequential Decision & Quantifying Risk using Dynamic Programming

We can then easily solve the problem using Dynamic Programming as a 4-stage problem, working backward, starting from Stage 4, and then Stage 3, Stage 2, and finally Stage 1. This is illustrated with the network diagram in Figure 2.

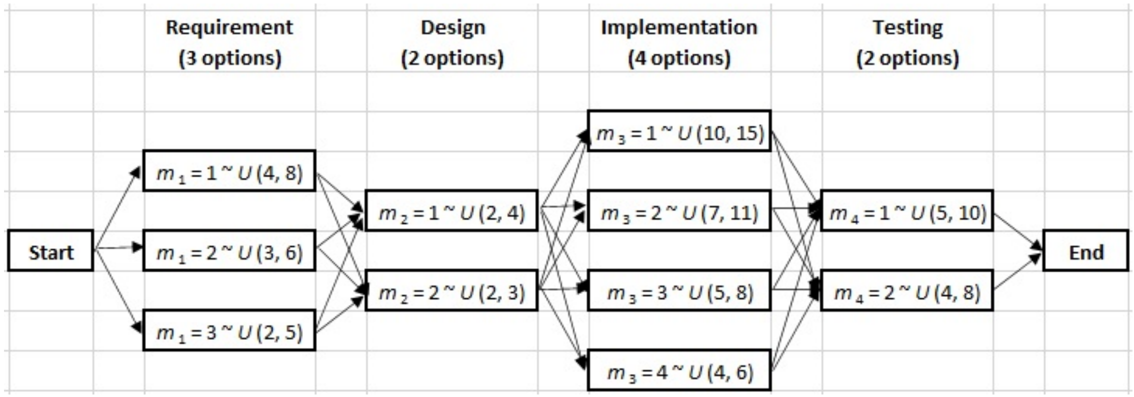

Figure 2. Network representation of DP as 4-Stage problem

We should point out that Dynamic Programming is the most natural optimization technique for sequential decision problems like what we have.

4. Numerical Computation and Discussion of Project Risk

There are few observations to make in Figure 2. First, we can calculate the project completion time as $3 \times 2 \times 4 \times 2 = 48$ different convolutions of random variables using Theorem 2.2 in Killmann & von Collani (2001).

Another alternative to solve the problem using Dynamic Programming is to work backward. Given the due date of $T_{max} = 20$ days, we can just use discritization of time interval with $\Delta t = 1$ day. Notice that from Table 1, if we look at only using 1 resource for all stages, the project completion time is given by $lb = 21$ and $ub = 42$. Similarly, we can see that using all resources $(m_1 = 3, m_2 = 2, m_3 = 4, m_4 = 2)$ will give $lb = 12$ and $ub = 22$ to complete the project.

Given that due-date of the project is $T_{max} = 20$, we can start the calculation backward as follows:

Stage 4 (Testing):

Assume that t_4 is the start of doing testing (*i.e.*, the starting time of Stage 4). Since we discretize with $\Delta t = 1$ day, we want to point out that the work content will be made simpler as well to be daily. For example: $W_{4,2} = 2 \times U(4{,}8)$ man-days is approximated with 2 persons with duration p_4 that is discretized into 4, 5, 6, 7, & 8 days – each day having probability = 0.2 $(= \frac{1}{5})$. The total cost (as well as optimal decision – in green color) for Stage 4 for various starting time t_4 is given in Table 2.

Table 2. Expected total cost & optimal decision for Stage 4 with various starting time t_4

c_T	c_4	m_4	lb_4	ub_4	t_4	24	23	22	21	20	19	18	17
5	1.2	1	5	10	$E[TC(t_4)] =$	66.50	61.50	56.50	51.50	46.50	41.50	36.50	31.50
5	1.2	2	4	8	$E[TC(t_4)] =$	64.40	59.40	54.40	49.40	44.40	39.40	34.40	29.40

c_T	c_4	m_4	lb_4	ub_4	t_4	16.00	15.00	14.00	13.00	12.00	11.00	10.00	9.00
5	1.2	1	5	10	$E[TC(t_4)] =$	26.50	21.50	17.33	14.00	11.50	9.83	9.00	9.00
5	1.2	2	4	8	$E[TC(t_4)] =$	24.40	20.40	17.40	15.40	14.40	14.40	14.40	14.40

There are several interesting points that are worth mentioning from Table 2. We denote that $j = 1$ and $j + 1 = 2$, and first, notice that when the starting time $t_4 \leq 12\ (= T_{max} - ub_{4,j+1})$, using $m_4 = 2$ persons to do testing will cost a constant $= 14.4 = 2 \times 1.2 \times \frac{4+8}{2}$ because it will never be late (recall that with 2 persons, the duration for testing will be at worst 8 days. Similarly, when $t_4 \leq 10\ (= T_{max} - ub_{4,1})$, using $m_4 = 1$ SQE, the total cost is a constant $9 = 1 \times 1.2 \times \frac{5+10}{2}$. Hence, given the linear nature of resource cost, it is clear that when $t_4 \leq 10$, it is better to use 1 SQE to do the testing. The next interesting question becomes, when it becomes beneficial to use 2 SQEs?

When $t_4 > 16\ (= T_{max} - lb_{4,j+1})$, is clear that using all resources (2 SQEs) will not be able to avoid tardiness since the best is just 4 days. Hence, the costs, that we have to compare, are:

- Using 1 SQE: $TC = c_4 \times 1 \times \left(\frac{5+10}{2}\right) + c_T\left(16 + \frac{5+10}{2} - 20\right) = 7.5c_4 + 3.5c_T$, *vs*
- Using 2 SQEs: $TC = c_4 \times 2 \times \left(\frac{4+8}{2}\right) + c_T\left(16 + \frac{4+8}{2} - 20\right) = 12c_4 + 2c_T$.

This realization can be written as in Lemma 1.

Lemma 1:

Given the condition [1] for Stage 4, *i.e.*, $W_{4,j+1} \sim n_{4,j} p_{4,j\to j+1} U_{4,j}\ \forall j = 1, 2, \dots, m_4$ and $TC = c_4 n_{4,j} U_{4,j} + c_T \times max\{0, t - T_{max}\}$, necessary condition to switch resource utilization from $n_{4,j}$ to $n_{4,j+1}\ \forall t_4 > T_{max} - lb_{4,j+1}$ is given by:

$$\frac{c_T}{c_4} > \frac{(n_{4,j+1}E[U_{4,j+1}] - n_{4,j}E[U_{4,j}])}{(E[U_{4,j+1}] - E[U_{4,j}])}$$

Proof:

Straightforward as explained in the previous paragraph.■

We would like to remark that the condition in Lemma 1 above implies that $W_{4,j} \sim n_j U_{4,j}(lb_j, ub_j)$ and $W_{4,j+1} \sim n_{j+1} U_{4,j}(lb_{j+1}, ub_{j+1})$ with $lb_{j+1} < lb_j$ and $ub_{j+1} < ub_j$. With inequality [4], we can see that the cost of tardiness $c_T > \frac{(n_{4,j+1}E[U_{4,j+1}] - n_{4,j}E[U_{4,j}])}{(E[U_{4,j}] - E[U_{4,j+1}])} c_4 =$

$\frac{2\times6-1\times7.5}{7.5-6}c_4 = 3c_4 = 3.6$ in order for optimal decision to switch using more resources $\forall t_4 > T_{max} - lb_{4,j+1}$.

With inequality [4] in Lemma 1, we know that $\forall t_4 \leq T_{max} - ub_{4,1}$, the optimal decision is to use 1 SQE, and $\forall t_4 \geq T_{max} - lb_{4,m_4}$, the optimal decision is to use all m_4 resources. In Stage 4, we just need to enumerate $t_4 \in [T_{max} - ub_{4,1}, T_{max} - lb_{4,m_4}]$. This helps to ease the curse of dimensionality in Dynamic Programming, we don't need to enumerate all possible values for t_4. In theory, we don't need to consider when $t_4 < 10$ since we know that using 1 SQE will be sufficient to complete the task. In general, the switching point to use 2 SQEs (more resources) will happen when inequality [4] holds which is in our case since $c_T = 5 > 3.6$ satisfy the condition.

Now, we can expand further to consider if $m_4 > 2$. Luckily, we can still use inequality [4] since it is general enough. Consider the following example in Table 5 (the only change is 1 additional resource in Testing (stage 4).

Table 3. Equivalent to Table 1, except 1 additional resource in Testing stage

		$m_j=1$		$m_j=2$		$m_j=3$		$m_j=4$			
Activity *i*	m_i	$lb_{(i,1)}$	$ub_{(i,1)}$	$lb_{(i,2)}$	$ub_{(i,2)}$	$lb_{(i,3)}$	$ub_{(i,3)}$	$lb_{(i,4)}$	$ub_{(i,4)}$	$p_{i,j\to j+1}$	c_i
Requirement	$m_1=3$	4	8	3	6	2	5	NA	NA	$p_{1,j\to j+1}\approx 0.75$	1.7
Design	$m_2=2$	2	4	2	3	NA	NA	NA	NA	$p_{2,j\to j+1}\approx 0.80$	1.5
Implementation	$m_3=4$	10	15	7	11	5	8	4	6	$p_{3,j\to j+1}\approx 0.70$	1.4
Testing	$m_4=3$	5	10	4	8	3	6	NA	NA	$p_{4,j\to j+1}\approx 0.80$	1.2
	Completion Time =	21	37	16	28	10	22	11	20		

With more available resources in Stage 4, we need to check $C_2^3 = 3$ inequalities from [4], namely:

- $\frac{c_T}{c_4} > \frac{(n_{4,2}E[U_{4,2}]-n_{4,1}E[U_{4,1}])}{(E[U_{4,1}]-E[U_{4,2}])} = 3$ (as before),
- $\frac{c_T}{c_4} > \frac{(n_{4,3}E[U_{4,3}]-n_{4,2}E[U_{4,2}])}{(E[U_{4,2}]-E[U_{4,3}])} = \frac{(3\times4.5-2\times6)}{(6-4.5)} = 1$, and
- $\frac{c_T}{c_4} > \frac{(n_{4,3}E[U_{4,3}]-n_{4,1}E[U_{4,1}])}{(E[U_{4,1}]-E[U_{4,3}])} = \frac{(3\times4.5-1\times7.5)}{(7.5-4.5)} = 2$

Given that our condition meet, then we have the following expected total cost and optimal decision in Stage 4 for various values of t_4.

Table 4. Expected total cost & optimal decision for Stage 4 with various starting time t_4 with $m_4 = 3$

c_T	c_4	m_4	lb_4	ub_4	t_4	24	23	22	21	20	19	18	17
5	1.2	1	5	10	$E[TC(t_4)]=$	66.50	61.50	56.50	51.50	46.50	41.50	36.50	31.50
5	1.2	2	4	8	$E[TC(t_4)]=$	64.40	59.40	54.40	49.40	44.40	39.40	34.40	29.40
5	1.2	3	3	6	$E[TC(t_4)]=$	58.70	53.70	48.70	43.70	38.70	33.70	28.70	23.70

c_T	c_4	m_4	lb_4	ub_4	t_4	16	15	14	13	12	11	10	9
5	1.2	1	5	10	$E[TC(t_4)]=$	26.50	21.50	17.33	14.00	11.50	9.83	9.00	9.00
5	1.2	2	4	8	$E[TC(t_4)]=$	24.40	20.40	17.40	15.40	14.40	14.40	14.40	14.40
5	1.2	3	3	6	$E[TC(t_4)]=$	19.95	17.45	16.20	16.20	16.20	16.20	16.20	16.20

Notice that managerially the expected total cost reduces by the additional SQE resource since we can go all out if we realize that the completion is late to avoid the steep tardiness penalty. However, this extra resource provide extra computation since the $lb_{4,3}$ becomes lower. Hence, $T_{max} - lb_{4,3} = 20 - 3 = 17$ becomes bigger, *i.e.*, enumeration space $t_4 \in [T_{max} - ub_{4,1}, T_{max} - lb_{4,3}] = [10,17]$.

Stage 3 (Implementation)

Using the same reasoning to cut the enumeration, we just need to enumerate for $t_3 \in [4, 12]$. The reason is very simple. From Table 2, we can see that the fastest that Stage 1 & 2 can finish is $lb_{1,3} = 2$ and $lb_{2,2} = 2$. This defines the minimum starting time $t_3 = 4$. Similarly, given that the maximum upper bound $ub_{3,1} = 15$ and we have calculated the previous $E[TC(t_4)]$ for $t_4 = 24$, we can see that we just need to calculate for $t_3 = 24 - 15 = 9$. Table 5 has the complete enumeration for functional equation [5]:

$$E[TC(t_3)] = \left\{\frac{1}{(ub_3 - lb_3 + 1)}\left(c_3 m_3 \sum_{t_3=lb_3}^{ub_3} t_3 + E[TC(t_4)]\right)\right\}$$

with the transformation function $t_{4,j} = t_{3,j} + d_{3,j}$ where: $d_{3,j}$ is the random variable duration to finish Implementation (Stage 3) using level j of resources. For clarify, with $j = 3$ (*i.e.*, using 3 Software Engineers, we have $d_{3,3} = 5, 6, 7, and\ 8$ days, each having probability of $\frac{1}{4} = 0.25$). We tabulated the result in Table 5 (and mark the optimal decision with green color).

It is very important to understand that in the functional equation [5] the second term, *i.e.*, $E[TC(t_4)]$ are independent from the value of m_3. It is just defined by the transformation function $t_{4,j} = t_{3,j} + d_{3,j}$. Therefore, we can take advantage of this property in our computation implementation.

Table 5. Expected total cost & optimal decision for Stage 3 with various starting time t_3

c_3	m_3	lb_3	ub_3	t_3	12	11	10	9	8	7	6	5	4	3
1.4	1	10	15	$E[TC(t_3)] =$	69.40	64.40	59.40	54.40	49.57	45.06	40.82	37.01	33.74	31.18
1.4	2	7	11	$E[TC(t_3)] =$	59.60	54.80	50.39	46.31	42.73	39.81	37.53	35.87	34.87	34.37
1.4	3	5	8	$E[TC(t_3)] =$	50.18	46.33	43.11	40.47	38.38	37.13	36.51	36.30	34.05	31.80
1.4	4	4	6	$E[TC(t_3)] =$	45.24	42.28	39.78	38.11	37.28	37.00	37.00	34.00	54.83	77.33

It is interesting to see that the optimal decision for Stage 3 is almost at the extreme, namely: when the starting point of implementation t_3 can be done early (*e.g.*, day 3 or 4), we can use 1 Software Engineer to do the job because we still have a chance to catch up if it slips. However, when the starting time for Stage 3 is tight, *i.e.*, $t_3 \geq 7$, the decision is to use up all (*i.e.*, 4 SEs). It is interesting to note the combinatoric nature of the problem when $t_3 = 6$, then the best decision is to use 3 SEs (rather than 4 SEs). Of course, this is also dependent on the cost structure. To illustrate this point, consider the following Table 8 if the unit cost of software engineer is changed to become $c_3 = 1.75$, we have a different optimal decision for $t_3 = 7$.

Table 6. Different optimal decision for Stage 3 with various starting time t_3 *if* $c_3 = 1.75$

c_3	m_3	lb_3	ub_3	t_3	12	11	10	9	8	7	6	5	4	3
1.75	1	10	15	$E[TC(t_3)] =$	73.78	68.78	63.78	58.78	53.94	49.43	45.20	41.38	38.12	35.55
1.75	2	7	11	$E[TC(t_3)] =$	65.90	61.10	56.69	52.61	49.03	46.11	43.83	42.17	41.17	40.67
1.75	3	5	8	$E[TC(t_3)] =$	57.01	53.16	49.93	47.29	45.21	43.96	43.33	43.13	40.88	38.63
1.75	4	4	6	$E[TC(t_3)] =$	52.24	49.28	46.78	45.11	44.28	44.00	44.00	41.00	61.83	84.33

Stage 2 (Design) & Stage 1 (Requirement)

Lastly, using the same reasoning, we can obtain the sequential optimal decision for Stage 2 (Design) and Stage 1 (Requirement Collection) respectively to be in Table 7.

Table 7. Expected TC & optimal decision for Stage 2 & Stage 1 with various starting times t_2 and $t_1 = 0$

c_2	m_3	lb_2	ub_2	t_2	8	7	6	5	4	3	2
1.5	1	2	4	$E[TC(t_2)] =$	46.93	44.56	42.89	41.96	41.43	40.34	39.25
1.5	2	2	3	$E[TC(t_2)] =$	48.53	46.44	45.19	44.64	44.25	42.75	41.37

c_1	m_1	lb_1	ub_1	t_1	0
1.7	1	4	8	$E[TC(t_1)] =$	51.35
1.7	2	3	6	$E[TC(t_1)] =$	54.55
1.7	3	2	5	$E[TC(t_1)] =$	44.96

The minimum expected cost is 44.96, and the overall optimal decision is to collect all requirements at the beginning as fast as possible using all 3 PMs so that the requirement becomes very clear and solid. Then, it is sufficient to use 1 Software Architect to design the feature that Aplikasi Super needs for the Web Dashboard to do the Business Process. Of course, depending on the completion timing of Stage 3 & various cost structures, a certain number of SEs (1, 3, or 4 SEs) may be needed to finish the Web Dashboard feature. Lastly, and again depending on the starting time of Stage 4, the best decision is either utilize all 2 available SETs (if t_4 is rather late) or just use 1 SET if there is enough time to do testing.

This backward Sequential Decision approach using Dynamic Programming provides a clear blueprint on what a manager should do at each stage depending on the outcome of the previous stage.

Of course, as we can see in the numerical computation of Stage 4 and Stage 3, some flexibility may be needed to adjust the number of resources in order to complete the project with minimal cost. This optimal decision may be better achieved if team members have the flexibility (agility) to switch roles rather than static on one particular role.

5. Conclusion and Further Research

We have demonstrated a very simple backward sequential decision process using Dynamic Programming to address one of the biggest drawbacks of Waterfall methodology, namely: risk management. With this technique, managers will have a blueprint on how to react for every stage of decision. We used Waterfall methodology successfully in the development of 3 different new features at Aplikasi Super.

Starting planning optimization backward with the Testing stage gives the decision maker a clear picture about the trade-off that needs to happen. Hence, provide a clear approach to manage the risk as the project progresses. The numerical example also illustrates the need to capture all requirements as clearly as possible using all available resources so that the later stages can be optimized better. Furthermore, flexibility in the resources for the later stage is very crucial to anticipate the possibility of using different resources. Therefore, it is recommended that a company has this type of flexible resources that can take different roles at different stages.

The modeling of sequential decisions here is very generic. It can accommodate the fact that adding more resources can create a more chaotic environment (Brooks' law) by proper modeling of the work content.

It is well known, though, that this DP approach suffers from what is called the curse of dimensionality. However, in this particular sequential process, by iterating over the starting time, we can reduce some enumeration by considering forward as well as backward processes and some conditions. Similarly, noting that our backward optimization optimizes on starting time, we can reduce the storage requirement.

For the future, a more formal way to reduce the computational of DP may be needed. This will be an interesting subject to be pursued.

References

Felix Torres and Herbert D. Benington, presentation at Navy Mathematical Computing Advisory Panel, Symposium on advanced programming methods for digital computers, Office of

Naval Research, Dept. of the Navy, Washington, D.C., 28 – 29 June 1956. Reprinted with foreword in H. D. Benington (1983), "Production of Large Computer Programs", *Annals of the History of Computing*, Vol. 5, No. 4, October 1983, pp. 350 – 361. https://doi.org/10.1109/MAHC.1983.10102.

Hall, Philip (1927), "The Distribution of Means for Samples of Size N Drawn from a Population in which the Variate Takes Values Between 0 and 1, All Such Values Being Equally Probable", *Biometrika*, Vol. 19, No. 3/4., pp. 240 – 245. https://doi.org/10.2307/2331961.

Irwin, J.O. (1927), "On the Frequency Distribution of the Means of Samples from a Population Having any Law of Frequency with Finite Moments, with Special Reference to Pearson's Type II", *Biometrika*, Vol. 19, No. 3/4., pp. 225 – 239. https://doi.org/10.2307/2331960.

Olds, E. G. (1952), "A Note on the Convolution of Uniform Distributions", *The Annals of Mathematical Statistics*, *23*(2), pp. 282 – 285. http://www.jstor.org/stable/2236455.

Wagner, H. M., & Whitin, T. M. (1958), "Dynamic Version of the Economic Lot Size Model", *Management Science*, 5(1), pp. 89 – 96. https://doi.org/10.1287/mnsc.5.1.89.

Royce, Winston (1970), "Managing the Development of Large Software Systems", *Proceedings of IEEE WESCON*, 26 (August), pp. 1 – 9. Republish in: *Proceedings of the 9th international conference on Software Engineering*, IEEE Computer Society Press, Washington, DC, USA (1987), pp. 328 – 338.

Brooks, Frederick P. (1975), *The Mythical Man-Month: Essays on Software Engineering*, Reading, Mass., Addison-Wesley Pub. Co.

Thomas E Bell and Thomas A Thayer (1976), "Software Requirements: Are They Really a Problem?", in *Proceedings of the 2nd international conference on Software engineering (ICSE '76)*, IEEE Computer Society Press, Washington, DC, USA, pp. 61 – 68.

Sei Ueda, Sohei Okada, Hajime Sato, & Kuno Shimizu (1994), "Distribution of the Sum of Uniform Random Variables with Different Ranges", *SUT Journal of Mathematics*, Vol. 30, No. 1, pp. 65 – 73.

Elmaghraby, Salah. (2005), "On the Fallacy of Averages in Project Risk Management", European Journal of Operational Research, Vol 165, https://doi.org/10.1016/j.ejor.2004.04.003.

Killmann, Frank and von Collani, Elart (2001), "A Note on the Convolution of the Uniform and Related Distributions and Their Use in Quality Control", *Economic Quality Control*, Vol. 16, No. 1, pp. 17 – 41. https://doi.org/10.1515/EQC.2001.17.

Van Casteren, Wilfred. (2017). The Waterfall Model and the Agile Methodologies : A comparison by project characteristics. https://doi.org/10.13140/RG.2.2.36825.72805.

Hillier, F. S., & Lieberman, G. J. (2021), *Introduction to operations research*, 11th Ed, New York: McGraw-Hill.

Theo Thesing, Carsten Feldmann, Martin Burchardt (2021), "Agile versus Waterfall Project Management: Decision Model for Selecting the Appropriate Approach to a Project", Procedia Computer Science, Volume 181, pp. 746 – 756. https://doi.org/10.1016/j.procs.2021.01.227.

Chandran, K., & Das Aundhe, M. (2022), "Agile or Waterfall Development: The Clementon Company Dilemma", *Journal of Information Technology Teaching Cases*, Vol. 12 No. 1, pp. 8 – 15. https://doi.org/10.1177/2043886919870544.