A Novel Approach for Optimization and Scheduling using SDLC (OSSDLC)

Vendhan Duraisamy
Assistant Professor, Information Technology
Kamaraj College of Engineering and Technology, Virudhunagar, India

Pradhiba Selvarani M
Assistant Professor, Computer Science and Engineering
Kamaraj College of Engineering and Technology, Virudhunagar, India

Abstract - An innovative approach to solve the optimization and scheduling problems using a single technique called Software Development Life Cycle (SDLC). SDLC is taken as the base technique to complete any problem. Here, we applied SDLC to solve the problems in two different domains like, (i) arranging jobs to schedule processes and (ii) Knapsack problem, to enhance optimization. We compared and analyzed different methods to solve the same problem by SDLC, where all five stages play an important role in solving the problem. First, in requirement gathering, identif y the possible inputs to the system. Second, analyze various algorithms that are applicable and not applicable with the help of Low Level Design (LLD) and solve the problem using High Level Design (HLD). Third, implement the appropriate algorithm that yields the better result. Fourth, test the algorithmic steps and procedure with various inputs. Fifth, maintain the tested algorithm to solve the problems in future effectively and efficiently to yield better performance. This proposed technique would let anyone to learn easily on how to apply the various scheduling mechanisms (from Operating Systems), algorithm design techniques (from Design and Analysis of Algorithms) to the real world problem and to generate optimal solutions to the same by mapping them with appropriate SDLC models (from Software Engineering). Hence, there is an integrated way of learning and achieving the course outcome.

Index Terms - Optimization, Scheduling, SDLC

I. INTRODUCTION

A software development methodology refers to the framework that is used to plan, manage, and control the process of developing an information system [2]. It involves complex processes using a hierarchy of activities and their documentation. There are number of models available in SDLC, each describing approaches to a range of tasks or activities that take place during the process [1]. There are many types of algorithms for scheduling the CPU jobs. Following are the measurement for good scheduling algorithms [4]. The high efficient CPU scheduler depends on design of the high quality scheduling algorithms which suits the scheduling jobs [5].

Maximize the CPU utilization Maximize the efficiency Maximize the response time

There are three natural possibilities under Greedy selection policy of DAA [7],

Policy 1- Choose the lightest remaining item, and take as much of it as can fit.
Policy 2 - Choose the most profitable remaining item, and take as much of it as can fit.
Policy 3 - Choose the item with the highest price per unit weight (P_i/W_i) and take as much of it as can fit**.**

I.1 Problem Statement

A contest is planned to be conducted as summer special for the farmers. Three farmers were selected to participate in it, based on the lucky draw. There exists a weighing machine that can hold up to 100 kg. All the three farmers have different baskets filled with variety of fruits. Each basket has its own weight value and yields separate profit value as well. All the farmers will be given one chance each. The farmer must place the baskets into the weighing machine in a way that it does not exceed 100 Kg. If entire basket doesn't fit into the machine causing overload, he can take samples from other fruit bags that he possess. Farmers must not exchange their fruits with other farmer.

TABLE I. Profit & Weight Values

FRUITS →	Apple		Banana		Cherry		Durian		Orange	
Farmers ↓	W_1	P_1	W_2	P_2	W_3	P_3	W_4	P_4	W_5	P_5
Farmer 1	40	100	30	60	60	150	40	80	20	40
Farmer 2	70	420	30	120	40	400	50	250	-	-
Farmer 3	50	250	60	180	60	480	20	200	-	-

Where,
W_i → Weight of Fruit Basket (in Kg.), P_i → Profit of Fruit Basket (in Rs.)

TABLE II. Question Set - OS & DAA

Requirement Collection	What are the requirements needed to solve the above mentioned problem?
Analysis / Design	Mention the different types of process scheduling mechanisms and algorithm design techniques available. Answer the following and provide proper justification: a. Which are all the mechanisms or techniques applicable and not applicable to this problem? b. Calculate and analyze which mechanisms or techniques gives the maximum profit value and why?
Coding / Computation	Solve the problem to get the maximum profit value using the correct mechanism or technique.
Testing	Check whether the steps and procedures of the mechanism or technique are correct or not?
Implementation	Implement the mechanism or technique that gives the maximum profit value effectively and efficiently.

II. EXISTING METHOD

All the scheduling and optimization problems are solved individually. Each problem has its own algorithm and procedure. There was no technique to analyze the various algorithms applicable to solve the same problem. The Greedy Approach does not always results in an optimal solution and dynamic programming method having time complexity of O(nW) where, 'n' is number of items and 'W' is the capacity of the knapsack [6].

III. PROPOSED METHOD

To provide an efficient solution by applying procedures of operating systems and the various problem solving techniques and map it to the Software Development Life Cycle as in "Fig 1". The challenge is to decide which model should be selected to provide a particular set of functionalities under certain circumstances [3].

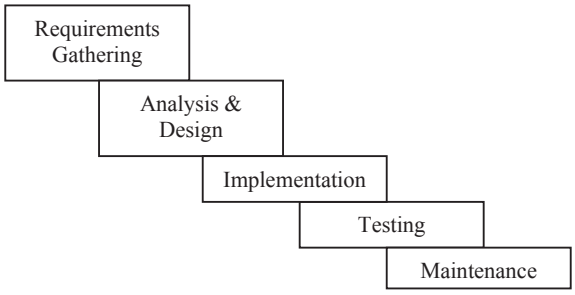


Fig. 1. Software Development Life Cycle

III.1 Solution Set - OS

Correlate the Operating System concept of Process Scheduling to detect the fruits with maximum profit.

1. Requirements Gathering

TABLE III. Fruits Profit & Weight

Fruits	Fruit Weight (Kg.)	Profit (Rs.)
Apple	40	100
Banana	30	60
Cherry	60	150

Durian	40	80
Orange	20	40

Threshold value of the weighing machine = 100 Kg

2. Techniques Available

a. Analysis LLD

First Come First Serve (FCFS), Shortest Job First (SJF), Priority, Round Robin

i. Applicable Techniques

First Come First Serve (FCFS), Shortest Job First (NP), Priority (NP), Round Robin

ii. Not Applicable Techniques

Preemptive SJF & Preemptive Priority
The problem doesn't depend on arrival time and hence the concept of Preemption is not applicable.

b. Calculate and Analysis HLD

Let us analyze the net profit obtained in each fruit by plotting through Gantt chart.

FCFS - First Come First Serve

It is the simplest scheduling algorithm. The processes are dispatched according to their arrival time on the queue [8].

Apple	Banana

0 40 70

Profit obtained with Apple = Rs.100 Profit obtained with Banana = Rs.60
Net Profit = 100 + 60 = **Rs.160**

SJF (NP) - Shortest Job First

Best approach to minimize the waiting time. The job with the shortest burst time is executed first [9].

Orange	Banana	Apple

0 20 50 90

Profit obtained with Orange = Rs.40 Profit obtained with Banana = Rs.60
Profit obtained with Apple = Rs.100 Net Profit = 40 + 60 + 100 = **Rs.200**

Priority (NP)

Each process is assigned a priority. Process with highest priority is to be executed first and so on [10].
Here priority is based on profit/weight value. The priority can be assigned based on the formula,

Calculate P_i / W_i and arrange it in descending order.

$A / W_1 = 100 / 40 = 2.5$ $B / W_2 = 60 / 30 = 2$
$C / W_3 = 150 / 60 = 2.5$ $D / W_4 = 80 / 40 = 2$
$E / W_5 = 40 / 20 = 2$

Hence, the fruits can be re-arranged in the following order according to their priority as follows,

2.5 2.5 2 2 2
A C B D O

Apple	Cherry

0 40 100

Profit obtained with Cherry = Rs.100 Profit obtained with Durian = Rs.150
Net Profit = 100 + 150 = **Rs.250**

Round Robin (Weight sample 10 kg)

Each process is provided a fix time to execute called quantum. Once a process is executed for given time period. Process is pre-empted and other process executes for given time period [11].

Profit obtained with Apple = (100 * 20) / 40 = 50
Profit obtained with Banana = (60 * 20) / 30 = 40
Profit obtained with Cherry = (150 * 20) / 60 = 50
Profit obtained with Durian = (80 * 20) / 40 = 40
Profit obtained with Orange = (40 * 20) / 20 = 40

Net Profit = 50 + 40 + 50 + 40 + 40 = **Rs.220**

III.2 Solution Set - PST

1. Requirements Gathering

KB = 100

$P_1 = 100$ $P_2 = 60$ $P_3 = 150$ $P_4 = 80$ $P_5 = 40$
$W_1 = 40$ $W_2 = 30$ $W_3 = 60$ $W_4 = 40$ $W_5 = 20$

Where, KB - Knapsack Bag, P - Profit value, W - Weight value

2. Techniques Available

a. Analysis LLD

Exhaustive Search, Divide and Conquer, Greedy Technique, Dynamic Programming, Branch & Bound, Backtracking & Approximation Algorithms

i. Applicable Techniques

Exhaustive Search, Greedy Technique, Dynamic Programming, Branch & Bound

The objective function is to maximize the profit value
It holds optimal solution and principle of optimality

ii. Not Applicable Techniques

Divide and Conquer - It works by recursively breaking down a problem into two or more sub-problems of the same type. It mainly applies to sorting, searching and recursive type of problems.

Backtracking - It applies to non - optimization problems.

b. Calculate and Analysis HLD

An optimization problem means either maximizing or minimizing the given objective function.

Brute Force

It is trial and error method. It is a straight forward approach to solve the problem.

TABLE IV. Total Weight and Profit Values with Subset

Sl. No.	Subset	Total Weight	Total Profit
1	{ø}	0	0
2	{1}	40	100
3	{2}	30	60
4	{3}	60	150
5	{4}	40	80
6	{5}	20	40
7	{1,2}	70	160
8	{1,3}	100	250
9	{1,4}	80	180
10	{1,5}	60	140
11	{2,3}	90	210
12	{2,4}	70	140
13	{2,5}	50	100
14	{3,4}	100	230
15	{3,5}	80	190
16	{4,5}	60	140

Sl. No.	Subset	Total Weight	Total Profit
17	{1,2,3}	130 (>100)	Infeasible
18	{1,2,4}	110 (>100)	Infeasible
19	{1,2,5}	90	200
20	{1,3,4}	140 (>100)	Infeasible
21	{1,3,5}	120 (>100)	Infeasible
22	{1,4,5}	100	220
23	{2,3,4}	130 (>100)	Infeasible
24	{2,3,5}	110 (>100)	Infeasible
25	{2,4,5}	130 (>100)	Infeasible
26	{3,4,5}	120 (>100)	Infeasible
27	{1,2,3,4}	170 (>100)	Infeasible
28	{1,2,3,5}	150 (>100)	Infeasible
29	{1,2,4,5}	130 (>100)	Infeasible
30	{1,3,4,5}	160 (>100)	Infeasible
31	{2,3,4,5}	150 (>100)	Infeasible
32	{1,2,3,4,5}	190 (>100)	Infeasible

Greedy Technique for Discrete

It solves an optimization problem by iteratively building a solution.

Step 1 - Calculate P_i / W_i values
Step 2 - Arrange the inputs in descending order of P_i/W_i values
Step 3 - Repeat until no item is left in the sorted list
Step 4 - If the current item fits into knapsack then place the item into knapsack and proceed to the next item otherwise just proceed to the next item

$P_1 / W_1 = 100 / 40 = X_1 = 2.5$ $P_2 / W_2 = 60 / 30 = X_2 = 2$
$P_3 / W_3 = 150 / 60 = X_3 = 2.5$ $P_4 / W_4 = 80 / 40 = X_4 = 2$
$P_5 / W_5 = 40 / 20 = X_5 = 2$

Descending Order = 2.5 2.5 2 2 2 = X_1 X_3 X_2 X_4 X_5

Profit Values = 100 150 60 80 40
Weight Values = 40 60 30 40 20

$X_1 = 1$ KB = 100 - 40 = 60
$X_3 = 1$ KB = 60 - 60 = 0
$X_2 = 0$
$X_4 = 0$
$X_5 = 0$

Optimal Solution = 100 * (1) + 60 * (0) + 150 * (1) + 80 * (0) + 40 * (0)
= 100 + 0 + 150 + 0 + 0 = **250**

Greedy Technique for Continuous

The goal is to fill the knapsack with fractional amount of weight values to maximize the profit value. It has a feasible solution and optimal solution [14].

Step 1, Step 2, Step 3 is same as greedy technique for discrete method.
Step 4 - If the current item fits into knapsack then place the item into knapsack and proceed to the next item otherwise take the largest function of the item to fill knapsack capacity and stop

Descending Order = 2.5 2.5 2 2 2 = X_1 X_3 X_2 X_4 X_5

Profit Values = 100 150 60 80 40
Weight Values = 40 60 30 40 20

$X_1 = 1$ KB = 100 - 40 = 60
$X_3 = 1$ KB = 60 - 60 = 0
$X_2 = 0$
$X_4 = 0$

$X_5 = 0$

Optimal Solution = 100 * (1) + 60 * (0) + 150 * (1) + 80 * (0) + 40 * (0)
= 100 + 0 + 150 + 0 + 0 = **250**

Dynamic Programming

It solves each of the sub problems only once and stores the result in the form of a table. It holds the principle of optimality [13].

Step 1 - Set the initial condition values C[i,0]=0 and C[0,j]=0
Step 2 - Table can be filled row by row or column by column
Step 3 - C[i, j] - It is the maximum value in the previous row and same column.

If C [5, 100] > C [4, 100] 250 > 240
No
If C [4, 100] > C [3, 100] 250 > 250
No
If C [3, 100] > C [2, 100] 250 >160
Yes
100 - 60 = 40
If C [2, 40] > C [1, 40] 100 > 100
No
If C [1, 40] > C [0, 40] 100 > 0
Yes
40 - 40 = 0
C [0, 0] Stop

TABLE V. Dynamic Programming Calculation

i	0	10	20	30	40	50	60	70	80	90	100
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	100	100	100	100	100	100	100
2	0	0	0	60	100	100	100	160	160	160	160
3	0	0	0	60	100	100	100	160	160	210	250
4	0	0	0	60	100	100	100	160	180	210	250
5	0	0	0	60	100	100	140	160	180	210	**250**

The objects included are $W_3 = 60$ $W_1 = 40$

Optimal Solution = 100 * (1) + 60 * (0) + 150 * (1) + 80 * (0) + 40 * (0)
= 100 + 0 + 150 + 0 + 0 = **250**

Branch and Bound

It is a systematic list of candidate solutions by means of state space search and modeling the solution as a tree [12].

Step 1 - Calculate P_i / W_i and arrange it in descending order.

Step 2- Record the total weight w, total profit p and some upper bound value.

Descending Order = 2.5 2.5 2 2 2
$= X_1$ X_3 X_2 X_4 X_5
Profit Values = 100 150 60 80 40
Weight Values = 40 60 30 40 20

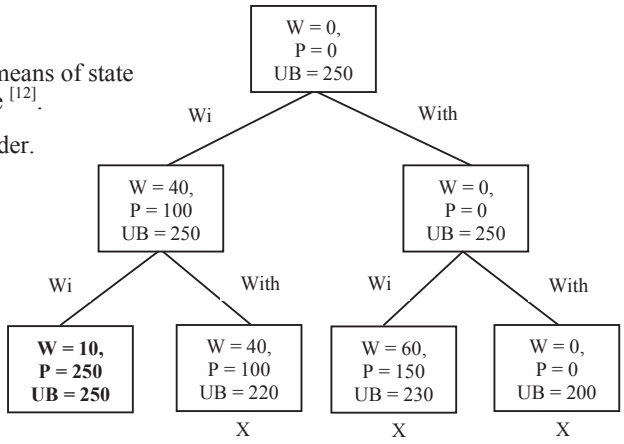


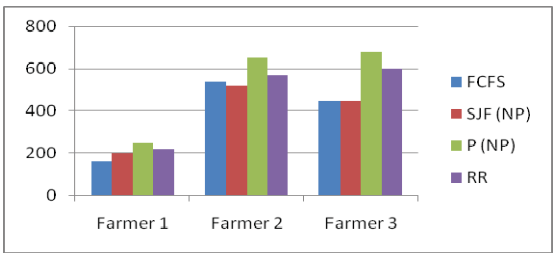
Fig 2. Weight, Profit & Upper Bound Values

IV. EXPERIMENTAL RESULTS AND ANALYSIS

TABLE V. Solution Set - OS

Techniques	Optimal Solution		
	Farmer 1 (Set-1)	Farmer 2 (Set-2)	Farmer 3 (Set-3)
FCFS	160	540	450
SJF (NP)	200	520	450
P (NP)	250	650	680
RR	220	570	600

Fig 3. Comparison Chart for OS

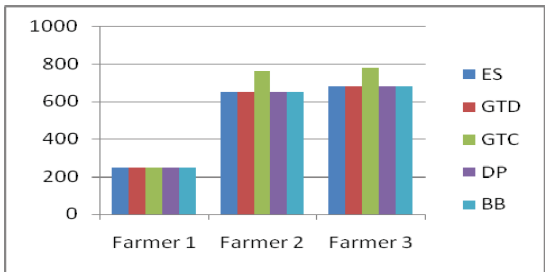


In all the three cases no other scheduling algorithm is having the optimal solution (maximum profit value) greater than priority based scheduling algorithm as in "Fig 3".

TABLE V. Solution Set - DAA

Techniques	Optimal Solution		
	Farmer 1 (Set-1)	Farmer 2 (Set-2)	Farmer 3 (Set-3)
ES	250	650	680
GTD	250	650	680
GTC	250	760	780
DP	250	650	680
BB	250	650	680

Fig 4. Comparison Chart for DAA



In all the three cases no other algorithm technique is having the optimal solution (maximum profit value) greater than greedy technique for continuous knapsack algorithm as in "Fig 4".

V. CONCLUSION

In all the mechanisms, weight values must be less than or equal to the threshold weight value set. Then find sum of the profit values of the objects which are inserted into the queue. Priority mechanism gives the maximum profit value. The remaining process scheduling mechanisms yields profit value less than or equal to Priority (NP). In all the techniques weight values must be less than or equal to the knapsack bag capacity value.

Then find sum of the profit values of the objects which are inserted into the knapsack bag capacity. Greedy technique for continuous knapsack problem gives the maximum profit value. The remaining algorithm design techniques are discrete or 0/1 knapsack problem type and yields profit value less than or equal to continuous.

VI. FUTURE ENHANCEMENTS

The pedagogy can be imbibed into this novel approach in the future by classifying the methodology into three phases like (i) Attain phase (Focus about Introduction to concepts - No mastery expected at this level), (ii) Align phase (Focus about depth in concepts - Students need to have enough familiarity with the subject and (iii) Integrate phase (Focus about depth in alignment - Participants expected to be masters). The students can be assessed based on conduction of activities including assignments, quizzes, Wikis, Forums, peer reviews [15].

REFERENCES

[1] T.Bhuvaneswari, S.Prabaharan, *"A Survey on Software Life Cycle Models"*, International Journal of Computer Science and Mobile Computing, Vol. 2, Issue. 5, May 2013, pg. 262 – 267.

[2] Apoorva Mishra and Deepti Dubey, *"A Comparative Study of Different Software Development Life Cycle Models in Different Scenarios"*, International Journal of Advance Research in Computer Science and Management Studies, Volume 1, Issue 5, October 2013, ISSN: 2321-7782.

[3] Vanshika Rastogi, *"Software Development Life Cycle Models – Comparison, Consequences"*, International Journal of Computer Science and Information Technologies, Vol. 6 (1), 2015, 168-172.

[4] Imran Qureshi, *"CPU Scheduling Algorithms: A Survey"*, International Journal on Advanced Networking and Applications, Volume: 05, Issue: 04, Pages: 1968-1973 (2014) ISSN: 0975-0290.

[5] Pushpraj Singh, Vinod Singh, Anjani Pandey, *"Analysis and Comparison of CPU Scheduling Algorithms"*, International Journal of Emerging Technology and Advanced Engineering, Volume 4, Issue 2, January 2014.

[6] Vikas Thada and Shivali Dhaka, "*Genetic Algorithm based Approach to Solve Non Fractional (0/1) Knapsack Optimization Problem*", International Journal of Computer Applications (0975-8887) Volume 100-No.15, August 2014.

[7] Rajinder Singh, *"A Case Study on Shifting Items with Max Profit in KNAPSACK with GREEDY Approach"*, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 11, November 2015 ISSN: 2277 128X.

[8] Vikram Singh and Tirlok Gabba, *"Comparative study of processes scheduling algorithms using simulator"*,International Journal of Computing and Business Research (IJCBR), ISSN: 2229-6166, Volume 4 Issue 2 May 2013.

[9] Ankur Bharadwaj, Sewa Devi and Gaurav, *"Comparative Study of SchedulingAlgorithms in Operating System"*, International Journal of Computers and Distributed Systems, ISSN: 2278-5183, Vol. No. 3, IssueI, April-May 2013.

[10] Karan Sukhija, Panja, Naveen Aggarwal and Manish Jindal, *"An Optimized Approach to CPU Scheduling Algorithm – Min-max"*, Journal of Emerging Technologies in Web Intelligence, Vol. 6, No.4, November 2014.

[11] Manish Kumar, Mishra and Dr. Faizur Rashid, *"An Improved Round Robin CPU Scheduling Algorithm with Varying Time Quantum"*, International Journal of Computer Science, Engineering and Applications, Vol. 4, No.4, August 2014.

[12] Archit Rastogi, Ankur Kumar Shrivastava, Nitisha Payal and Ramander Singh, *"A Proposed Solution to Travelling Salesman Problem using Branch and Bound"*, International Journal of Computer Applications (0975-8887), Volume 65 – No.5, March 2013.

[13] Feixue Yan and Feng Bai, *"Application of Dynamic Programming Model in Stock Portfolio-under the Background of the Subprime Mortgage Crisis"*, International Journal of Business and Management, Vol. 4, No. 3, March 2009.

[14] Navneet Kaur, *"Outlier Detection Using a New Hybrid Approach Based on Group Weighted K-Mean and Greedy Method on Mixed Dataset"*, International Journal of Advanced Research in Computer Science and Software Engineering, ISSN: 2277 128X, Volume 5, Issue 8, August 2015.

[15] Evrim Baran, *"Connect, Participate and Learn: Transforming Pedagogies in Higher Education"*, Bulletin of the IEEE Technical Committee on Learning Technology, Volume 15, Number 1, January 2013.