A SIMULATION MODEL BASED ON MARKOV DECISION PROCESS TO ASSESS SOFTWARE QUALITY

Ömer Korkmaz
HAVELSAN A. Ş.,
Eskişehir yolu 7. Km. 06520 Ankara, Turkey,
+ 90–312-2195797, Email: okorkmaz@havelsan.com.tr

İbrahim Akman †
Department of Computer Engineering,
Atilim University, 06836 Incek-Golbasi Ankara Turkey
+ 90-312-5868348, +90-312-5868091, Email: akman@atilim.edu.tr

Sofiya Ostrovska
Department of Mathematics
Atilim University, 06836 Incek-Golbasi Ankara Turkey
+90-312-5868211, +90-312-5868091, Email: ostrovska@atilim.edu.tr

Abstract

This article proposes a simulation model utilizing Markov Decision Process (MDP) for quality assessment of Software Development Process (SDP). The MDP is based on a qualification model, which contains project architecture, team assignment strategy, Software Quality Assurance (SQA) system construction strategy, and qualification actions selected in the SQA system. The proposed approach has been demonstrated using a sample software project taken from the literature. The results prove the proposed approach to be capable of assessing alternative strategies at earlier stages of SDP.

Keywords: Software Quality, Software Quality Assurance, Modeling, Markov Decision Process, Simulation

1. INTRODUCTION

The properties of software products show important differences from other industrial products (Galin 2004) and make the assessment of software product quality more difficult. Most of the proposed approaches generally benchmark the performance of software components and do not consider dynamics of SDP, in which case, analysts' intuition and experience plays important role and have significant impact on the desired judgment on the quality. This is unfortunate because SDP dynamics are important (Warren et al. 1992) and modelling these dynamics should be a routine part. In order to consider the system dynamics effectively, SDP must include techniques that facilitate the effectiveness of the verification of the target process. Traditional stochastic simulation is one of such powerful techniques available to those responsible for the development of software systems. The stochastic simulation models not only models the dynamics of architectural design but also increases the accuracy of early estimates of quality. Consequently, it saves effort at an earlier stage of

† Corresponding author

development, rather than waiting until after the system is implemented (Hall et al., 2002). However, stochastic simulation has not been a common practice for SDP and is still in its infancy (Pfahl, 2002).

This article proposes a simulation model based on the stochastic nature of SDP to represent the dynamics of the development process such as project architecture, SQA construction strategy, qualification actions, and team assignment strategy. The proposed approach has the advantage of testing different theoretical performance targets of different phases at earlier stages and can be tailored to the special features of the SQA system of software projects because it is parametrized. It also increases the accuracy of early estimates of the quality. The proposed approach has been demonstrated using a case study taken from the literature (Padberg 2002).

2. QUALIFICATION MODEL

The approach here uses a qualification model containing important dynamics of a software development project, such as task assignments, the skill level of project staff, the SQA components, and rework caused by detected errors. As mentioned earlier, the qualification model is stochastic and takes into account the uncertainty of the occurrence of events during the sub-phases of the software project development. In order to include the dynamics in the qualification model, the project architecture, base probabilities, qualification actions, team assignment strategy, SQA construction strategy, and their corresponding assumptions need to be explained (Figure 1).

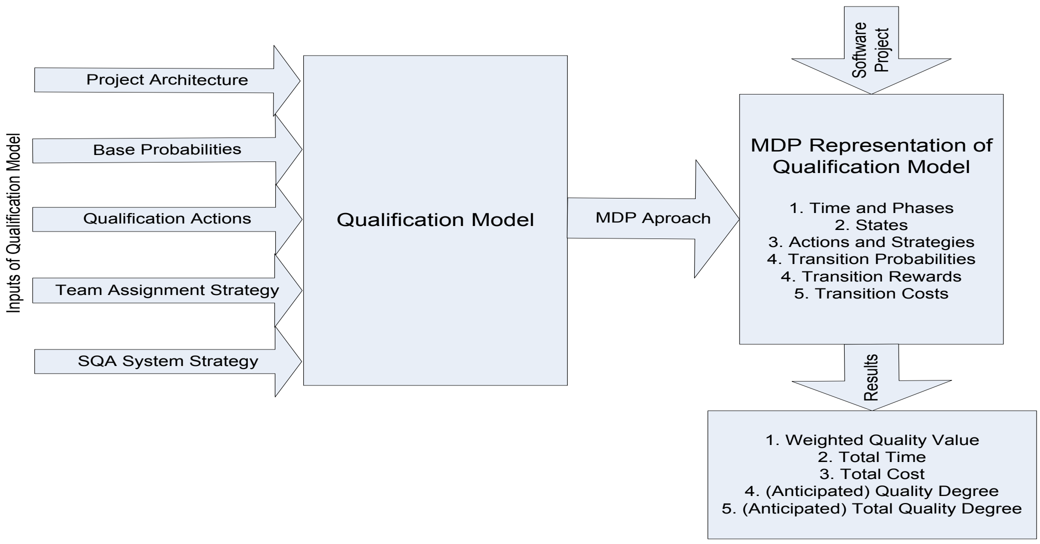

Figure 1: System modelling

2.1 Project Architecture

The architecture of a software development project has a significant impact on the complexity and quality and is taken as one of the factors of the qualification model. The proposed approach assumes that project architecture is determined by the components of the project, components to be developed simultaneously and the development order of the components. Additionally, the project components are assumed to be classified according to their size and complexity level (Table 1).

Table 1: Classes of software project components and teams

Class No	Component		Team			
	Estimated Size	Estimated Complexity Level	Productvity Level	Error Making Level	Cost Level	Unit staffing Cost
1	Small	Low	Low	Low	7	0.850
2	Small	Average	Low	Average	8	0.825
3	Small	High	Low	High	9	0.800
4	Average	Low	Average	Low	3	0.950
5	Average	Average	Average	Average	4	0.925
6	Average	High	Average	High	6	0.875
7	Large	Low	High	Low	1	1.000
8	Large	Average	High	Average	2	0.975
9	Large	High	High	High	5	0.900

2.2 Base Probabilities

The SDP presents a stochastic environment since the events occur with certain probabilities during SDP lifecycle. Therefore, qualification model includes prediction of these probabilities, which are called base probabilities. The base probabilities reflect the characteristics of the project components, the SQA components and the teams. In other words, they show how the project teams make progress in working/reworking on the project components, and determine how the SQA components behave on the project components as follows:

1. $P\left(A_{TEAM\_k}^{COMP\_r}(t)\right)$ and $P\left(B_{TEAM\_k}^{COMP\_r}(t)\right)$ specify the probability that the team k will complete its work on the project component r without errors and with at least one error after having worked on it for t time units respectively.
2. $P\left(C_{SQA\_COMP\_k}^{COMP\_r}(t)\right)$, $P\left(D_{SQA\_COMP\_k}^{COMP\_r}(t)\right)$ and $P\left(E_{SQA\_COMP\_k}^{COMP\_r}(t)\right)$ specify the probability that the SQA component k will detect all the errors, least one error - but not all – and no errors on the project component r after having worked on it for t time units respectively.

3. $P\left(F_{TEAM\_k}^{COMP\_r}(t)\right)$, $P\left(G_{TEAM\_k}^{COMP\_r}(t)\right)$ and $P\left(H_{TEAM\_k}^{COMP\_r}(t)\right)$ specify the probability that the team k will remove all the errors, at least one error - but not all – and no errors from the project component r after having worked on it for t time units respectively.

The base probabilities, in the model, represent the probability of transition from one state to proceeding one and depend on the size and complexity level of the components, and productivity and error-making level of the teams. The project components are assumed to be classified into nine classes with respect to their estimated size and complexity level as given in Table 1. The skill and cost level for a team are assumed to be determined by the corresponding productivity and error making levels (Table 1). The skill level (productivity level - error making level) is used to calculate base probabilities although team cost level is for the cost.

2.3 Qualification Actions

Qualification actions take place only at the end of phases (or sub-phases) in which the development team completes its work on the project component. General reviews, formal design reviews, tests are examples of qualification actions. The SQA component applied during the qualification action is assumed to affect the result of the action. This means the characteristics of the qualification actions can be modeled using the base probabilities using the metrics database of the organization.

2.4 Team Assignment Strategy

Team assignment strategy has important impact on the quality level of SDP and, therefore, is included as one of the components of the qualification model. The proposed qualification model assumes that in a software project there may be more than one team with different skill levels and a project may consist of more than one component. The skill level can be represented by the productivity and error making levels given in Table 1 and predicted using the base probabilities.

2.5 SQA System Construction Strategy

Construction strategy is the way of choosing components to implement an SQA system in an organization. The SQA system should be tailored to reflect the project's internal dynamics. It should take important characteristics and quality factors of the project, quality policies and objectives of the organization and department, and staff skill level of the project department into account (Pressman, 2000; Galin, 2004).

3. MDP MODEL

The MDP model is based on time and phases, project states, actions and strategies, transition probabilities, and reward and cost functions for the design phase of SDP life cycle. The MDP uses a set of states S, a set of actions A and a reward function. A subset of actions is available for each state and there is probability distribution for each state. In consideration of the proposed model, all MDP's are finite, that is both sets S and A are finite.

3.1 Time and Phases

The time is taken into account in the model in terms of base probabilities and the term phase has been used to indicate the progress of the project with respect to the time spent.

In the workload model, a software project is assumed to advance through a sequence of phases, which may consist of some sub-phases. By definition, a phase ends when a team finishes its work, or rework, on a project product, or when the application of an SQA component on a project product is finalized. This allows us to use a discrete-time finite-horizon MDP as a mathematical model. In this study, the software development process has been adopted as a sequential decision problem, in which the set of actions, rewards, cost, and transition probabilities depend only on the current state of the system and the current action being performed.

3.2 States

In the MDP, the system moves from one state to another randomly. The state of a system is a parameter, or a set of parameters, that describes the system. The state § of software project changes at the end of each phase and consists of four components: a status vector, an assignment vector, a progress vector, and a countdown variable.

The status vector has one entity for each component of the software project and an entity of the status vector is defined as the status of a component. During a software development phase, the status of a project component may have one of the following values: Not yet developed (NYD); Under development (UD); Developed with no error (DWNE); Developed with error(s) (DWE); Under qualification action (UQA); Qualified by detecting no error (QBDNE); Qualified by detecting some errors (QBDSE); Qualified by detecting all errors (QBDAE); Under rework (UR); Reworked by removing no error detected (RBRNED); Reworked by removing some errors detected (RBRSED); Reworked by removing all errors detected (RBRAED); Completed (COMP) and Cancelled (CANCEL).

The assignment vector defines which component is assigned to which team and has one entity for each component of the software project,. An entity of the assignment vector is defined as the team ID (team 1, 2, etc.) which has worked most recently, or is still working, on the component. An entity is equal to 0 if none of the project teams has worked on the corresponding component yet.

The progress vector has one entity for each component of the software project too. An entity of the progress vector is defined as the time that spent while working on the component in a phase. In the current phase, if the work is completed on the r-th component, the entity of the assignment vector will show ∞. Similarly, if a team is assigned to the r-th component while no work has started on it yet i.e. 0).

The countdown variable shows the time left for the completion of the project which is denoted by T.

The state of a project component r can be defined at the end of a phase. The initial state of a project is defined by status vector =NDY, assignment vector=0, progress vector=0, countdown variable=T, where T>0. The final state means that all the project components are completed, and that no new team will be assigned to any component (i.e. for each component r, progress vector= ∞, status vector = COMP and assignment vector=team_ID). The project development also finishes when the project's deadline is exceeded, i.e., countdown variable <= 0. However, if the project has not been completed before its deadline, then the quality degree of the project is assumed to be zero, or any negative value, to indicate the effect of the previously-determined project development time.

3.3 Actions and Strategies

An action defines what will be done with regard to the project component at a given development phase. Actions may depend on the current state of the project as well as the current phase of the development process. The possible actions of the design phase of the are as follows.

1. "Component designing" sub-phase: Assigning a team to a component (d_1); Starting to design (d_2); Completing the design activities (d_3) and Canceling design activities (d_4).
2. "Design reviewing" sub-phase: Forming and assigning a review team (rd_1); Starting to review (rd_2); Completing review (rd_3) and Canceling review (rd_4).
3. "Design reworking" sub-phase: Assigning the team to rework (rw_1); Starting to rework (rw_2); Completing the rework (rw_3) and Canceling the rework (rw_4).

For the design phase, the action set A can be expressed as:

$$A = \{d_1, d_2, d_3, d_4, rd_1, rd_2, rd_3, rd_4, rw_1, rw_2, rw_3, rw_4\} \tag{1}$$

To form an assignment strategy, the components of the project are listed according to their priority level. Assignment strategies should keep all the project teams engaged during the project development. After the work on a component is completed, it is removed from the list and the next component on the list moves into the development process. If only one team is available, it is assigned to this component. If there is more than one team available, the team with the lowest ID is assigned to the component to be processed. The skill level of project teams and the classification of project components should be taken into account during the assignment of a team to a component.

3.4 Transition Probabilities

The transition probability represents the probability of a system to move from one state to another under a given action. For a given current state $§_i \in S$ and action $\alpha_i \in A_{§_i}$, the next state $§_{i+1}$ is not determined alone due to the stochastic nature of the selected dynamics, yet it occurs with the probability $P(§_i, \alpha_i; §_{i+1})$. Clearly, for a given state $§ \in S$ and action $\alpha \in A_§$, we have

$$\sum_{\varsigma} P(§, \alpha; \varsigma) = 1 \quad \text{where } \{\varsigma\} \text{ are all possible next states for } § \in S \tag{2}$$

Since in an MDP, the transition probabilities depend only on the current state and the action performed rather than those in the past, the probability that the project will take a particular path equals:

$$P(\psi_N) = \prod_{i=0}^{N-1} P(§_i, \alpha_i; §_{i+1}). \tag{3}$$

3.5 Quality in Terms of Reward and Cost

For each project component, the reward and cost functions have to be taken into account. To this end, a reward function (IR) has been used to determine the immediate reward (positive or negative). In the proposed model for the design phase, a positive immediate reward for each of the project components is

$IR(§_i, \alpha_i; §_{i+1}) = \{$

5	for $§_i$ = NYD, $\alpha_i = d_1$; $§_{i+1}$ = UD
90	for $§_i$ =UD , $\alpha_i = d_2$; $§_{i+1}$ = DWNE
10	for $§_i$ = UD, $\alpha_i = d_2$; $§_{i+1}$ = DWE
5	for $§_i$ =DWE , $\alpha_i = rd_1$; $§_{i+1}$ = UQA
0	for $§_i$ =UQA , $\alpha_i = rd_2$; $§_{i+1}$ = QBDNE
10	for $§_i$ =UQA , $\alpha_i = rd_2$; $§_{i+1}$ = QBDSE
20	for $§_i$ =UQA , $\alpha_i = rd_2$;$§_{i+1}$ = QBDAE
5	for $§_i$ = QBDSE, QBDAE, $\alpha_i = rw_1$, $§_{i+1}$ = UR
0	for $§_i$ =UR , $\alpha_i = rw_2$; $§_{i+1}$ = RBRNED
20	for $§_i$ =UR , $\alpha_i = rw_2$; $§_{i+1}$ = RBRSED
40	for $§_i$ =UR , $\alpha_i = rw_2$; $§_{i+1}$ = RBRAED
5	for $§_i$ = DWNE, $\alpha_i = d_3$; $§_i$ = QBDNE, $\alpha_i = rd_3$; $§_i$ = RBRNED, RBRSED, RBRAED, $\alpha_i = rw_3$; $§_{i+1}$ = COMP

-1000 for $§_i$ =UD, α_i=d_4; $§_i$ = DWE, UQA, α_i=rd_4; $§_i$ = QBDNE, QBDSE, UR, α_i=rw_4; $§_{i+1}$ = CANCEL} (4)

For a given path ψ, the total immediate reward is:

$$V_{immediate}(\psi) = \sum_{i=0}^{N-1} IR(§_i,\ \alpha_i;\ §_{i+1}) \tag{5}$$

We propose the use of the 'Weighted Quality Value' (WQV), as referred to in this paper, for each state action sequence ψ:

$$WQV(\psi) = \sum_{i=0}^{N-1} IR\left(§_i,\ \alpha_i;\ §_{i+1}\right).P(§_i,\ \alpha_i;\ §_{i+1}) \tag{6}$$

The transition cost is calculated as follows:

$C\left(§_i,\alpha_i;§_{i+1}\right) = \left(cost\_of\_team\right)*t$, where $t = §_i^c - §_{i+1}^c$ and cost_of_team is taken from Table 1

In our model, the quality of a path can be calculated by the quantity referred to in this paper as the 'Quality Degree' (QD):

$$QD(\psi) = WQV(\psi) / \sum_{i=0}^{N-1} C(§_i,\alpha_i;§_{i+1}) \tag{7}$$

However, Equation (7) does not lead to a meaningful comparison of team performance in a project since it shows the total quality degree only for a single path. To take into consideration all the possible paths for each team-component pair, we introduce the expected QD as:

$$QD_{expected} = \sum_{all\_paths\_\psi} QD\left(\psi\right).P\left(\psi\right) \tag{8}$$

4. SIMULATION MODEL

To get an overall conclusion about all possible strategies in a project, the quality degree (QD) of each strategy must be calculated (Sutton and Barto, 1998). Therefore, when the number of system states is high, simulation has been successfully used to calculate the mean value (mean quality degree) of a given strategy (Padberg 2000; Bertsekas 1996). The Mean quality degree for a given strategy ψ can be expressed as follows:

$$QD_{mean}\left(\psi_{§,N}\right) = \frac{1}{Y}\sum_{j=1}^{Y} QD\left(\psi_{§,N};j\right) \tag{9}$$

where $QD_{mean}\left(\psi_{§,N}\right)$ is the mean value (mean quality degree) of j-th sample trajectory starting from state § and having N steps to go. Y shows the number of sample trajectories used in the simulation. In the simulation, the mean quality degree of a sample trajectory can be calculated as follows (Padberg 2000; Bertsekas 1996):

$$QD_{mean}\left(\psi_{\S,N}\right) := QD_{mean}\left(\psi_{\S,N}\right) + \frac{1}{j}\left(QD_{mean}\left(\psi_{\S,N}; j\right) - QD_{mean}\left(\psi_{\S,N}\right)\right) \tag{10}$$

The mean quality degree $QD_{mean}\left(\psi_{\S,N}\right)$ is initialized with zero. Whenever passing each state on the simulation trajectory, the Eq. (10) is applied.

5. APPLICATION

To demonstrate the proposed model, a sample project has been selected from literature (Padberg 2002), and the model has been implemented by using ARENA® simulation tool according to the sample project.

5.1 Simulation Results

In the simulation model, the strategy followed during project development consists of two parts: The component development order and team assignment strategy. To analyze the results of the simulation, Average quality reward, average development cost, average development time, and average team utilization have been used. Outputs of each strategy have been measured and comparisons have been done by using Paired-t test of the Output Analyzer tool with either 95% or 99% confidence levels.

5.2 Effects of Deadline on Project Quality

The effects of the deadline on the quality of project have been analyzed by using two different scenarios. The execution time for each scenario is limited with a deadline. If a simulation run can not be accomplished before the deadline, the run will be aborted. For the 1st scenario, deadline has been selected so as to guarantee to stop some of the simulation runs. But for the 2nd scenario, deadline has been selected long enough to ensure all runs can be completed before deadline. The results of the 1st and 2nd scenarios for the strategy ABCD-1212 have been given in Table 3.

Table 3: Output values if deadline is exceeded

	deadline is exceeded				deadline is not exceeded			
Output	Ave.	Half Width	Min. Ave.	Max. Ave.	Ave.	Half Width	Min. Ave.	Max. Ave.
A- WQV	93.58	1.06	25.00	100.00	93.58	1.06	25.00	100.00
B- WQV	82.91	2.17	0.00	100.00	92.41	1.13	25.00	100.00
C- WQV	67.37	2.77	0.00	100.00	86.21	1.42	25.00	100.00
D- WQV	43.39	2.91	0.00	100.00	76.46	1.64	25.00	100.00

A - Cost	111.47	0.94	83.33	160.00	111.47	0.94	83.33	160.00
B - Cost	160.49	1.37	133.00	224.83	160.49	1.37	133.00	224.83
C - Cost	215.63	2.22	176.67	290.00	215.63	2.22	176.67	290.00
D - Cost	367.22	3.46	294.50	446.50	367.22	3.46	294.50	446.50
Total Cost	854.81	4.49	719.83	1069.50	854.81	4.49	719.83	1069.50
Total Dev. Time	168.68	< 1.18	137.00	206.04	168.68	< 1.18	137.00	206.04
Total WQV	287.24	4.65	25.00	400.00	348.65	2.59	175.00	400.00

Table 3 also shows the values when the deadline is not exceeded. Note that, the average WQV of components B, C and D is higher than that of deadline is exceeded, and the half width value is less to show the small variance between the replications. Cost and time values are the same in both cases.

The deadline is exceeded part of Table 3 declares a software project having a quality degree (unit WQV value per unit cost) of 287.24/854.81 after following the strategy of ABCD-1212. Table 3 also shows a quality degree of 348.65/854.81 for the same project and the same strategy. It is obvious that the 2nd case has a higher quality degree when the deadline is not exceeded.

5.3 The Best and Worst Strategies

Table 5 lists the best and worst strategies for each output measure and declares the observed values.

Table 5: The best and worst strategies

Output	Observed Value	Component Development Order	Team Assignment Strategy
Highest Average WQV	349.97	ABCD; ABDC; BACD; BADC	1221
Lowest Average WQV	346.43	ACBD; CABD	1122
Highest Average Development Cost	893.15	BCAD; CBAD	2212
Lowest Average Development Cost	771.31	DCAB; CDAB	1121
Longest Average Development Time (in days)	213.73	BCAD; CBAD	2212
Shortest Average Development Time (in days)	131.79	ADBC; BDAC; DABC; DBAC	1112

Best Average Team Utilization	11.10%	CDAB; DCAB	1112
Worst Average Team Utilization	69.33%	BCAD; CBAD	2212

The highest average WQV is 349.97 over 400. Note that, the maximum WQV for a project component is 100, and because the sample project has four components, total maximum WQV for it is 400. The highest WQV was satisfied by four development orders - ABCD, ABDC, BACD, and BADC- with the same team assignment strategy, 1221. It should be noted that in the simulation model, all project teams must be kept busy during development. So, while Team 1 develops Component A, Team 2 develops B, or vice versa. So, there is no difference between the development order AB and BA. The same situation is valid for the development of components C and D. Thus, those four development orders are the same for the strategy 1221. So, all of them have the same WQV as it is seen in Table 5.

The lowest average WQV is 346.43 over 400. This value is satisfied by two different development orders - ACBD and CABD - with the same team assignment strategy, 1122. Because, all project teams must be busy, the development order of AC and CA are the same.

The strategies ABCD-1221 and ACBD-1122 have been analyzed by using Output Analyzer tool to understand that the highest average WQV differs statistically from the lowest one. The result shows that those two strategies are statistically different from each other at 0.05 level (the 95% Confidence Interval), as the random input generator of the ARENA® uses different random input sets for each simulation run (Chapter 12 in (Kelton, Sadowski, and Sturrock 2007)). For further analysis, it can be seen that they are not different statistically at 0.01 level (the 99% Confidence Interval) by usingthe Paired-t Test, because the fail and success rates are defined as the same for Team 1 and 2. So, the quality level of a component developed by Team 1 is almost the same with that of the component developed by Team 2. The differences between quality levels of the strategies in Table 5 are as a result of the random inputs used in the simulation model. As a conclusion, since the average WQVs of all strategies are very close to each other, the quality level should not be taken into consideration while choosing the best and worst strategies for the sample project.

The highest average development cost is 893.15 units. The cost depends on not only the time but also the unit cost value of the teams who have spent the time. The highest cost value is satisfied by two development orders - BCAD and CBAD - the same for the team assignment strategy 2212. For both development orders, Team 2 develops Component B firstly (Team 1 develops only C) and then develops A and finally D. This means that two small components and the largest one are developed by Team 2. According to the probability distributions, Team 2 is slower than Team 1 and the former is cheaper than the latter. Keeping them in mind, it can be concluded that the strategies in which the slower team develops 3 components of the sample project may have the highest average development cost.

The lowest average cost is 771.31 units for two different development orders - CDAB and DCAB - the same for the team assignment strategy 1121. For both development orders, Team 1 develops Component D firstly (Team 2 develops only C) and then develops A and finally B. This means that two small components and the largest one are developed by Team 1. So, the strategies in which the faster team develops 3 components of the sample project may have the lowest average development cost.

The strategy BCAD-2212 and CDAB-1121 has been analyzed by using the Output Analyzer, using the Paired-t Test. The result shows that those two strategies are statistically different from each other at 0.05 level.

The longest average development time is 213.73 days. The development time depends on the probability distributions of the project teams. The longest development time value belongs to the development orders - BCAD and CBAD - the same for the team assignment strategy 2212. The strategy 2212 means that two small components and the largest component are developed by Team 2. Thus, the strategies in which the slower team develops three components may have the longest average development time.

The shortest average development time is 131.79 days. This value is satisfied by four different development orders - ADBC, BDAC, DABC, and DBAC - with the same team assignment strategy, 1112. Two small components and a large one are developed by Team 1. So, the strategies in which the faster team develops 3 components may have the shortest average development time.

The strategy BCAD-2212 and ADBC-1112 has been analyzed by using the Output Analyzer, using the Paired-t Test. The result shows that those two strategies are statistically different from each other at 0.05 level.

6. CONCLUSİONS

This work shows how simulation can be used to support project managers in finding a good strategy for achieving a high quality level for their projects. A simulation model based on the MDP model has been presented for only design phase for this purpose. The simulation model has been implemented in ARENA® simulation tool on a small hypothetical software project from (Padberg 2002) to analyze the performance of various strategies for software projects. Although, other methods using metrics and available standards can be used to assess the quality the simulation results show that proposed approach is capable of assessing alternative strategies at earlier stages of SDP. This constitute the most important advantage of the MDP method over other methods.

REFERENCES

Ambler, S. W. and Jeffries, R. (2001) Agile Modeling: Effective Practices for Extreme Programming and the Unified Process, 1st Edition, John Wiley & Sons.

Bertsekas, T. (1996) Dynamic Programming and Optimal Control I+II, Athena Scientific.

Boehm, B. W. (1981) Software Engineering Economics, Prentice-Hall, NJ, USA.

Galin, D. (2004) Software Quality Assurance – From theory to implementation, Pearson – Addison Wesley, England.

Hall T. et. al. (2002) Implementing Software Process Improvement: An Empirical Study, Software Process Improvement and Practice, 2002, 7, 3-15.

Haring, G. and Kotsis, G. (1995) Workload Modeling for Parallel Processing Systems. MASCOTS 1995, 8-12.

Kelton, W. D., Sadowski, R. P., Sturrock, D. T. (2007) Simulation with Arena, 4th Edition, McGraw Hill International, ISBN-13:978-0-07-110685-6.

Padberg, F. (2000) Towards Optimizing the Schedule of Software Projects with Respect to Development Time and Cost, International Software Process Simulation Modeling Workshop ProSim.

Padberg, F. (2002) A Discrete Simulation Model for Assessing Software Project Scheduling Policies, Software Process Improvement and Practice 7: 127-139.

Srivastava, A. K., Sharma, G. and Keeni, G. (2002) A Systematic Approach to Plan Inspection Process, The 7th International Symposium on Future Technology (ISFST), October 20-25, 2002 Xi'an, Wuhan, China.

Sutton, R. S. and Barto, A. G. (1998) Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA.