Design of Traditional/Hybrid Software Project Tracking Technique: State Space Approach

MANOJ KUMAR TYAGI[1], SRINIVASAN M.[2], L.S.S. REDDY[3]
[1]Electronics and Computer Engineering
[1,3]K. L. University
Vaddeswaram, Guntur, 522502 A.P.
INDIA
[1]manojkumar@kluniversity.in, [3]drlssreddy@kluniversity.in
[2]Electronics and Communication Engineering
Meerut Institute of Engineering and Technology
Meerut, 250005 U.P.
INDIA
[2]msrinivasan77@yahoo.com

Abstract: - Software projects are required to be tracked during their execution for controlling them. According to state space approach, the tracking problem leads us to have a project state transition model and project status model. A key factor in modeling software projects is to model the project with uncertainty involved in the parameters related to project state transition model and project status model. Traditional/Hybrid software project tracking technique is formulated, modeled with state space approach in plan-space and execution-space, and simulated using discrete event simulation. The uncertainty considered here is ontological that modeled as a normal distribution using an approximation method. The state space model consists of project-state transition equation and project-measurement equation, in plan-space, and it is formulated with Monte Carlo method in execution-space. The developed state space model is used to track status of the traditional/hybrid project. The project is executed with an iterative and incremental development process. With Monte Carlo simulation runs, simulation result shows the variation in ontological uncertainty iteration-wise, both individually and cumulatively, and the effect of uncertainty on project status, by showing project status in execution-space and plan-space. Besides, the project completion somewhere during the last iteration is shown with simulation.

Key-Words: - Ontological Uncertainty, Normal Distribution function, State-Space Model, Traditional Project, Hybrid Project, Monte Carlo Simulation, Discrete Event Simulation, Execution-Space, Plan-Space.

1 Introduction

How software projects are modeled for tracking by considering state-space view? This question is not taken into consideration during software project tracking. According to state-space view, software projects start with initial state, traverse through intermediary states, and move to final state. These have two views: plan-space view and execution-space view. Generally, the execution-space view differs with plan-space view; as software projects, consisting of both the technical and managerial activities, involve unknown factors including services, cost, schedule, size, productivity , lack of information, ambiguity, characteristics of project parties, tradeoffs between trust and control mechanisms, and varying agendas in different stages of the project life cycle ([1], [2]). As a general rule, uncertainty arises in any activity involving unknown factors, which affects the activity [3, p. 2-4].

Methods for software development, a set of practices for software development, are based on plan-based, agile, or hybrid software development philosophy ([4], [5]). Both agile and plan-driven philosophies have a home ground area of project characteristics in which each clearly works best, and where the other will have difficulties [6]. Hybrid approaches that combine both philosophies are feasible and necessary for projects that combine a mix of agile and plan-driven home ground characteristics ([6], [7]). In this paper, hybrid approach is derived with following requirements volatility phenomena partially, i.e., allowing the changes to the already specified requirements which were documented before starting the project execution; but not allowing the new requirements requested by customers during execution; while traditional approach does not follow requirements volatility phenomena.

In the past, considerable research has been conducted on improving the effectiveness of project management. However, much of the work has concentrated on developing and evaluating estimation and control tools used for software projects ([8], [9]), as opposed to model the software projects with uncertainty ([1], [10]-[12]) for tracking. Software project combines application domain knowledge, computer science, statistics, behavioral science, and human factor issues. One of the statistical research and education challenges is providing models with appropriate error distributions (uncertainty) for software projects [13]. This paper develops a state space model of software project tracking by considering state-space view. The developed state space model is used to track the status of traditional/hybrid project which is executed with an iterative and incremental development process [14].

2 Related Work

Software project modeling has been an important field of study since 1950's onwards. Traditional-models such as Work Breakdown Structure (WBS), Gantt Model, Critical Path Method (CPM) Model, PERT Model, AND-OR Graph Model, Petri Net Model, Stochastic Petri net-based Model, Design Net Model ([9],[15]) are used to support the operational issues. The Metrix model is a stochastic model for software project duration estimation using Monte Carlo simulation over an activity graph [16].The Project Monitoring with Stakeholders Model is based on the measurement information model defined by ISO/IEC 15939, and added stakeholder's (a purchaser and a developer) goal, key goal indicator(KGI),key performance indicator(KPI), corrective action, and check timing[17]. The Antipattern Bayesian Network Model provides the mathematical model of a project management antipattern and can be used to measure and handle uncertainty in mathematical terms [18]. The Software Project Tracking literature ([19], [20]) consists of GANTT Chart, SLIP Chart, TIMELINE Chart, CPM, PERT, COST Charts, S-Curve based methods such as Integrated Cost/Schedule/Work method and Earned Value Analysis are used for cost or schedule tracking.

Having gone through the literature, we found some issues which are not considered for tracking the software projects as

- The state-space view of software projects has not been considered for software project modeling.
- Little consideration has been given to project modeling for tracking with due consideration for uncertainty related to software development productivity, requirement's-specification document.
- There is a need of software project tracking technique designed with state-space approach and considering the above issues.

Here we have developed a state space model representing traditional/hybrid software project tracking technique considering uncertainty. The model is capable to track traditional/hybrid software project status in plan-space or execution-space.

3 Model Building

The traditional/hybrid software project tracking model is developed using the state space approach ([21], [22]). The model has developed based on a case study described in Mike Cohn's Book[23]. The case study here involves a mythical game-development company "Bomb Shelter Studios". In this study, a game termed 'Havannah' was developed with agile philosophy. Here, traditional software project tracking model considers uncertainty related to software development productivity; while hybrid software project tracking model considers uncertainty related to requirements volatility phenomenon, and software development productivity. Besides the definition of the model boundaries and model granularity, the most important design decisions were related to the typically observed behavior patterns ("reference mode") of development projects. The reference mode was defined by the dynamics of product evolution, i.e. a product is developed with iterative and incremental process model, i.e. each product increment implements certain types of requirements.

3.1 Dynamics of Product Evolution

The development of software product is done incrementally with equal periods (iteration). During each period one increment is developed.

3.2 Dynamics of Requirements Generation

At the beginning of each iteration, a fixed set of frozen-requirements selected from requirements specification document to start with is known. Here we assume that requirements volatility phenomena cause modification to leftover requirements for hybrid project, but not for traditional project. This leads new requirements to be generated during the iterations and updated to the requirement's specification document. These new requirements do not produce major changes in the software being built, but leads to more customer satisfaction. Typically, the number of new requirements shows a ceiling effect, i.e. the new requirements do not reflect modification or replacement of already implemented requirements.

4 Design of Traditional/Hybrid Software Project Tracking Technique

According to state-space approach, the software project tracking technique consists of project state-space and status-space. The project state-space and status-space are collection of project states and project statuses respectively. The software project starts with estimated initial-state, traverse through intermediate states, to final state in plan-space and execution-space. The software project plan-space and execution-space represent the project plan behavior and execution behaviour respectively. The project behavior is defined with project state and its status. As software project transits in its state-space, the project states are used to derive project statuses in status-space, during plan-space and execution-space. Generally, the project execution behavior differs with plan behavior due to uncertainty with project environment. The project is tracked to control.

Software project tracking techniques are designed with state-space approach ([21], [22]) in plan-space, and as well using Monte Carlo method [24] in execution-space. The state-space model, consisting of software project state-transition model and software project measurement model, is used to model a software project tracking technique. The following equations (1) and (2) are used to represent a software project tracking technique in plan-space and execution-space, respectively.

$$\begin{aligned}
&\textit{Software Project State Transition Model}\\
&\vec{S}_{t+1} = A \times \vec{S}_t\\
&\textit{Software Project Measurement Model}\\
&\vec{M}_t = H \times \vec{S}_t
\end{aligned} \tag{1}$$

$$\begin{aligned}
&\textit{Software Project State Transition Model}\\
&\vec{S}_{t+1} = A \times \vec{S}_t + \vec{N}_{t+1}\\
&\textit{Software Project Measurement Model}\\
&\vec{M}_t = H \times \vec{S}_t + \vec{V}_t
\end{aligned} \tag{2}$$

Where A and H represent a state-transition matrix and measurement transition matrix, respectively. $\vec{N}$ and $\vec{V}$ represent a project state noise (uncertainty) and measurement noise vector due to which software projects deviate from plan. $\vec{S}$ and $\vec{M}$ represent state vector and measurement vector, respectively. Above equations represent, the way a new state $\vec{S}_{t+1}$ is modeled as a linear combination of the previous state $\vec{S}_t$ in plan-space and both the previous state $\vec{S}_t$ and some project uncertainty $\vec{N}_{t+1}$ in execution-space; and how project status $\vec{M}_t$ is derived with the internal state $\vec{S}_t$.

5 State Space Modeling-A Novel Approach

During the life of software projects, their execution behavior differs with plan behavior due to uncertainty with project environment. The software project, during execution, is described as a stochastic-process ([25], [26]); which changes its state over the life of a project. Here, we are developing a new approach for modeling the software project tracking technique which considers state-space view for tracking traditional/hybrid software projects in plan-space and execution-space. The Monte Carlo method, which uses random numbers from a given probability distribution to compute something [24], is used to develop software project state transition model and project status model in execution-space.

5.1 State-Space Model: Traditional/Hybrid Software Project Tracking Technique Formulation

The software project behaves differently in plan-space and execution-space. This happens due to uncertainties with team-capability and requirements-specification document. The project is planned with uniform team-capability and requirements volatility phenomena. The team-capability represents the workload selected to be completed during iteration. Due to requirements volatility phenomena, there are changes to the requirements-specification document. The project-state is described with two parameters as P, and RR representing requirements selected to be completed during an iteration, and remaining requirements to be completed for project completion respectively. Fig.1 represents the project state at starting and ending point of (t+1)th iteration.

Fig. 1. Representation of Project-State at (t+1)th Iteration

The project is started with initial-state $[P_0, RR_0]^T=[16,132]^T$, where T represents matrix transpose operation. Team-Capability represented by P, depends on the working capability of the developers, used to select workload at starting of some iteration. There are three techniques-Historical Values, Make a Forecast, and Run an Iteration, for estimating it [23]. Here, it was estimated initially with historical values technique. It is uniform in plan-space and varying with ontological uncertainty in execution-space. It is affected with the factors such as schedule-pressure, communication/motivation, size-estimation, workforce-experience level, ([27]-[30]) etc in execution-space. Their net-effect on team-capability is a random variable ϵ_p representing ontological uncertainty. Team-Capability during last iteration is used for selecting the workload for present iteration.

$$P_{t+1} = P_t \quad \textit{Plan Space} \tag{3}$$
$$P_{t+1} = P_t + \varepsilon_{pt+1} \quad \textit{Execution Space} \tag{4}$$

Remaining-Requirements to be completed for project completion at the end of (t+1)th iteration, represented by RR_{t+1}, is calculated as follows:

(i)Team-Capability represented by P_t is used for selecting the workload to be completed for (t+1)th iteration, which are not affected with requirements-volatility phenomena.

(ii)Requirements left over during (t+1)th iteration, which are affected with requirements-volatility phenomena([31]-[34]), is (RR_t-P_t). "ϒ" is a stochastic variable controlling the changes occurring in the already specified requirements of the product. Estimating "ϒ" introduces the ontological uncertainty, represented by "ϵ_u", for remaining-requirements to be completed for project completion at the end of (t+1)th iteration in execution-space . New requirements generated due to requirements-volatility phenomena is ϒ(RR_t-P_t), which are to be added.

Therefore, remaining-requirements at the end of (t+1)th iteration is find out by adding: Requirements left over during (t+1)th iteration, which are affected with requirements-volatility phenomena, is (RR_t-P_t); and New requirements generated due to requirements-volatility phenomena is ϒ(RR_t-P_t); in plan-space and in addition to it adding ϵ_u ; in execution-space.

$$RR_{t+1} = \Upsilon(RR_t-P_t)+ (RR_t-P_t)$$

$$RR_{t+1} = -(\gamma+1)\times P_t + (\gamma+1)\times RR_t \quad \textit{Plan Space} \tag{5}$$
$$RR_{t+1} = -(\gamma+1)\times P_t + (\gamma+1)\times RR_t + \varepsilon_{ut+1} \quad \textit{Execution Space} \tag{6}$$

Ontological uncertainty (noise) is defined as inherent variability concerning the parameters used to model the system [35]. Normal noise is a popular choice which interferes with human decision-making [36]. The ontological uncertainty represented by random variables ϵ_p , and ϵ_u , has been modeled by using an approximation method, which derives normally distributed random numbers by summing several uniformly distributed random numbers x_i according to the following formula ([37],pp.158)

$$y = \frac{\sum_{i=1}^{k} x_i - \frac{k}{2}}{\left(\frac{k}{12}\right)^{1/2}} \qquad As\ k \to \infty \equiv (0:1) \tag{7}$$

Where "y" is random variable following a normal-distribution with mean "0" and standard deviation "1".

The state space model for software project tracking technique has been developed by using difference equations describing the dynamics of software project in plan-space. The developed model is used to represent traditional/hybrid project in plan-space. Equations (3) and (5) describe the project state transition in plan-space.

$$P_{t+1} = P_t$$
$$RR_{t+1} = -(\Upsilon+1)\, P_t + (\Upsilon+1)\, RR_t$$

$$P_{t+1} = 1\times P_t + 0\times RR_t \tag{8}$$
$$RR_{t+1} = -(\gamma+1)\times P_t + (\gamma+1)\times RR_t \tag{9}$$

The project status is described with remaining requirements to be completed for project completion, project velocity, effort remaining to complete the project, and team strength at time, t representing either start or end of some iteration. At time t, it relates to the state of the project as follows:

$$remainingrequirements = RR_t$$
$$projectvelocity = RR_{t-1} - RR_t$$

effortremaining = $\beta \times RR_t$,

Where β represents a factor which is used for converting remaining requirements into effort remaining. Assume a review of historical data indicates that the organizational productivity for product of this type is 4 story-points/ person-iteration. Based on a burdened labor rate of $ 2000 per person-iteration, the cost/story-points is 2000/4 = 500 $/ story-points. Then β is calculated as follows

β = (500 $/ story-points)/(2000 $/ person-iteration) =1/4 person-iteration/ story-points.

teamstrength = P_t

$$remainingrequirements = 0 \times P_t + 1 \times RR_t \quad (10)$$

$$projectvelocity = RR_{t-1} - RR_t \quad (11)$$

$$effortremaining = 0 \times P_t + \beta \times RR_t \quad (12)$$

$$teamstrength = 1 \times P_t + 0 \times RR_t \quad (13)$$

The software project state transition equation in matrix form for software project tracking technique in plan-space has been derived with equations (8), and (9); and software project measurement equation in matrix form for software project tracking technique in plan-space has been derived with equations (10), (12) and (13). Equation (14) represents a state space model for software project tracking technique in plan-space.

Software Project State Transition Equation

$$\begin{bmatrix} P_{t+1} \\ RR_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -(\gamma+1) & (\gamma+1) \end{bmatrix} \begin{bmatrix} P_t \\ RR_t \end{bmatrix}$$

Software Project Measurement Equation (14)

$$\begin{bmatrix} remainingrequirements \\ effortremaining \\ teamstrength \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \beta \\ 1 & 0 \end{bmatrix} \begin{bmatrix} P_t \\ RR_t \end{bmatrix}$$

Equations (11) and (14) are used to track the status of traditional/hybrid software project in plan-space.

The state space model for software project tracking technique has been developed by using difference equations describing the dynamics of software project in execution-space. The developed model is used to represent traditional/hybrid project in execution-space. Equations (4) and (6) describe the project state transition in execution-space.

$$P_{t+1} = P_t + \epsilon_{pt+1}$$
$$RR_{t+1} = -(\Upsilon+1) P_t + (\Upsilon+1) RR_t + \epsilon_{ut+1}$$

$$P_{t+1} = 1 \times P_t + 0 \times RR_t + \varepsilon_{pt+1} \quad (15)$$

$$RR_{t+1} = -(\gamma+1) \times P_t + (\gamma+1) \times RR_t + \varepsilon_{ut+1} \quad (16)$$

The project status is described with remaining requirements to be completed for project completion, project velocity, effort remaining to complete the project, and team strength at time, t representing either start or end of some iteration. At time t, it relates to the state of the project as follows:

remainingrequirements = RR_t

projectvelocity = $RR_{t-1} - RR_t$

effortremaining = $\beta \times RR_t + \epsilon_{\beta t}$,

Where β represents a factor, which is used for converting remaining requirements into effort remaining; and introduces ontological uncertainty ϵ_β for effortremaining related to status vector. The ontological uncertainty represented by random variable ϵ_β, has been modeled with equation (7). Assume a review of historical data indicates that the organizational productivity for product of this type is 4 story-points/ person-iteration. Based on a burdened labor rate of $ 2000 per person-iteration, the cost/story-points is 2000/4 = 500 $/ story-points. Then β is calculated as follows

β = (500 $/ story-points)/(2000 $/ person-iteration) =1/4 person-iteration/ story-points.

teamstrength = P_t

$$remainingrequirements = 0 \times P_t + 1 \times RR_t + 0 \times \varepsilon_{\beta t} \quad (17)$$

$$projectvelocity = RR_{t-1} - RR_t \quad (18)$$

$$effortremaining = 0 \times P_t + \beta \times RR_t + 1 \times \varepsilon_{\beta t} \quad (19)$$

$$teamstrength = 1 \times P_t + 0 \times RR_t + 0 \times \varepsilon_{\beta t} \quad (20)$$

The software project state transition equation in matrix form for software project tracking technique in execution-space has been derived with equations (15), and (16); and software project measurement equation in matrix form for software project tracking technique in execution-space has been derived with equations (17), (19), and (20). Equation (21) represents a state space model for software project tracking technique in execution-space.

Software Project State Transition Equation

$$\begin{bmatrix} P_{t+1} \\ RR_{t+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -(\gamma+1) & (\gamma+1) \end{bmatrix} \begin{bmatrix} P_t \\ RR_t \end{bmatrix} + \begin{bmatrix} \varepsilon_{pt+1} \\ \varepsilon_{ut+1} \end{bmatrix}$$

Software Project Measurement Equation (21)

$$\begin{bmatrix} remainingrequirements \\ effortremaining \\ teamstrength \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \beta \\ 1 & 0 \end{bmatrix} \begin{bmatrix} P_t \\ RR_t \end{bmatrix} + \begin{bmatrix} 0 \\ \varepsilon_{\beta t} \\ 0 \end{bmatrix}$$

Equations (18) and (21) are used to track the status of traditional/hybrid software project in execution-space.

The state space models for software project in plan-space and execution-space differs with ontological uncertainty represented by random variables ε_p, ε_β, and ε_u related to team -capability P, effortremaining, and remaining-requirements RR respectively. The random variables ε_p, ε_β, and ε_u follow normal probability distribution. In this paper, we have used the parameter Υ and ε_u to classify the tracking techniques as follows

- If Υ and ε_u are zero for all iterations then the model represents a traditional project tracking technique.
- If Υ is non-zero for all iterations then the model represents a hybrid project tracking technique.

6 Uncertainty Propagation

The software project in execution-space deviates from plan-space due to uncertainty involved with parameters team capability, effortremaining, and remaining requirements used for modeling. The uncertainty propagation is described iteration-wise, individually and cumulatively, in execution-space.

With equation (21), at time t=0, i.e. during first iteration

$$P_1 = P_0 + \epsilon_{p1}$$

$$RR_1 = -(\Upsilon+1)\, P_0 + (\Upsilon+1)\, RR_0 + \epsilon_{u1}$$

Clearly, uncertainty with team capability during first iteration,

$$IndividualDeviation\_TC_1 = \epsilon_{p1}$$

$$Deviation\_TC_1 = \epsilon_{p1}$$

Uncertainty with remaining requirements during first iteration,

$$Deviation\_RR_1 = \epsilon_{u1}$$

$$CumulativeDeviation\_RR_1 = Deviation\_RR_1$$

With equation (21), at time t=1, i.e. during second iteration

$$P_2 = P_1 + \epsilon_{p2}$$

$$RR_2 = -(\Upsilon+1)\, P_1 + (\Upsilon+1)\, RR_1 + \epsilon_{u2}$$

By expanding P_1 and RR_1, Uncertainty with team capability during second iteration,

$$Individual\ Deviation\_TC_2 = \epsilon_{p2}$$

$$Deviation\_TC_2 = \epsilon_{p1} + \epsilon_{p2}$$

Uncertainty with remaining requirements during second iteration,

$Deviation\_RR_2 = -(\Upsilon+1)\, \epsilon_{p1} + (\Upsilon+1)\, \epsilon_{u1} + \epsilon_{u2}$

$= -(\Upsilon+1)\, Deviation\_TC_1 + (\Upsilon+1)\, Deviation\_RR_1 + \epsilon_{u2}$

$CumulativeDeviation\_RR_2 = Deviation\_RR_1 + Deviation\_RR_2$

By generalizing, during $(t+1)^{th}$ iteration, Uncertainty with team capability, individually and cumulatively

$$IndividualDeviation\_TC_{t+1} = \varepsilon_{pt+1} \qquad (22)$$

$$Deviation\_TC_{t+1} = \varepsilon_{p1} + \varepsilon_{p2} + ... + \varepsilon_{pt+1} \qquad (23)$$

Uncertainty with remaining requirements, individually and cumulatively

$$Deviation\_RR_{t+1} = -(\gamma+1) \times Deviation\_TC_t + (\gamma + 1) \times Deviation\_RR_t + \varepsilon_{ut+1} \qquad (24)$$

$$Cumulative\ Deviation\_RR_{t+1} = Deviation\_RR_1 + ... + Deviation\_RR_{t+1} \qquad (25)$$

7 Simulation Results

The project is initialized with the state vector $[16, 132]^T$, where T represents matrix transpose operation. The software project changes its state because of the requirements completed by the developers. And the project status, at either start or end of an iteration, is determined with the state of the project. The state is determined by "Team_Capability" (story-points) representing the workload selected to be completed by the developers, and "Remaining_Requirements" (story-points) representing remaining requirements to be completed for project completion by developers. The status is determined by "Effort_Remaining" (person-iteration) effort needed to complete the project, "Team_Strength"(story-points/iteration) representing team capability to complete requirements per iteration, and "Project_Velocity" (story-points/iteration) representing completed requirements per iteration, "Remaining _ Requirements" (story-points) representing remaining requirements to be completed for project completion by developers.

With Monte Carlo runs, the deviation for Remaining_Requirements and Team_Capability has been shown iteration-wise, individually and cumulatively respectively; which affects the status of the project in execution- space. How project status, affected with uncertainty in remaining requirements, and Team_Capability is shown with simulation. The status of the project has been shown with planned and actual project execution at the end of each iteration.

7.1 Traditional Software Project Tracking

Fig. 2 shows the deviation for Remaining _Requirements and Team_Capability individually and cumulatively.

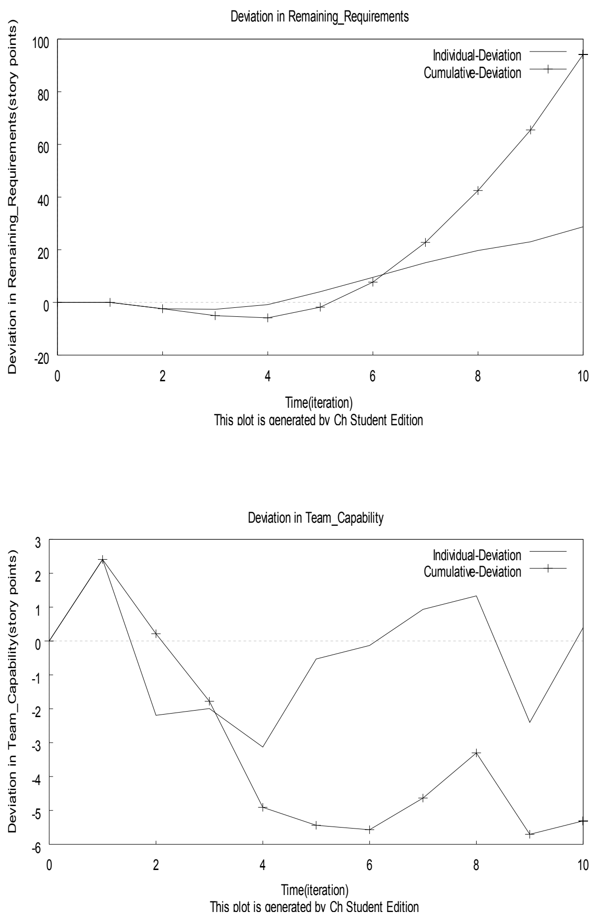
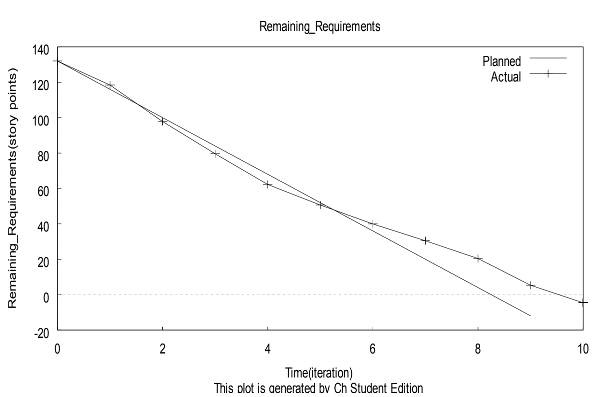


Fig. 2 Deviation for Remaining_Requirements and Team_Capability for Traditional Project

Fig. 3 shows the project status with Effort _Remaining, Team_Strength, Remaining _Requirements and Project_Velocity curves. The project has completed during 10^{th} iteration after planned duration during 9^{th} iteration.



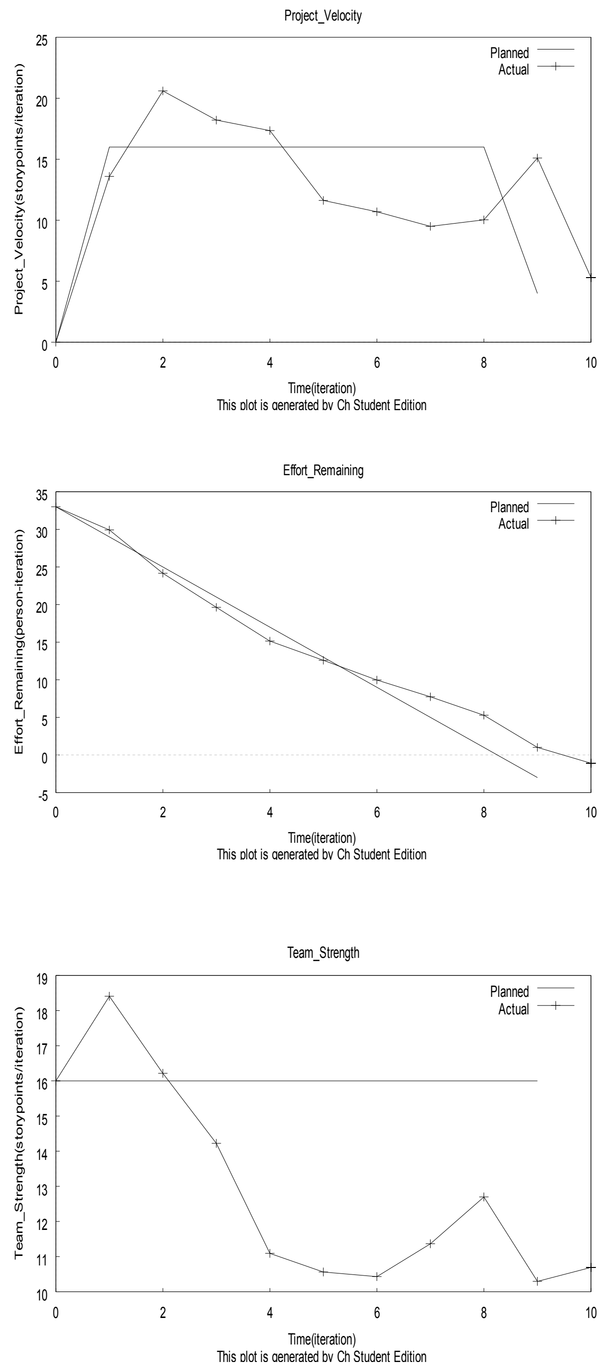


Fig. 3 Representation of Traditional Project Status

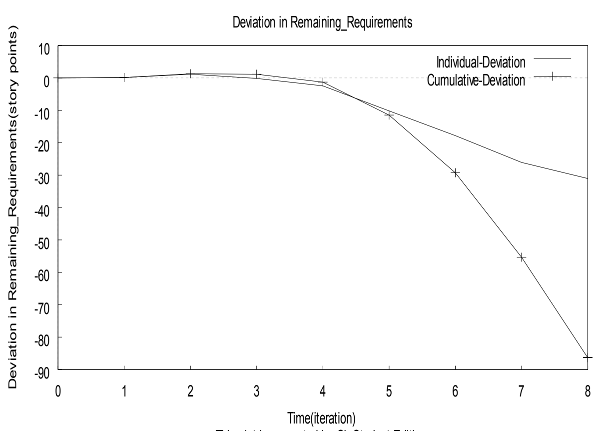
Table 1 shows the planned and actual project status by showing Remaining_Requirements, Effort_Remaining, Team_ Strength, and Project_Velocity at each iteration.

Planned Project Execution(Traditional Project)				
Iterati -on#	Remaining_ Requirements (story points)	Effort_Remai -ning (person iteration)	Team_Str -ength (story points/iter ation)	Project_Ve -locity (story points/itera tion)
0	132.000000	33.000000	16.000000	00.000000
1	116.000000	29.000000	16.000000	16.000000
2	100.000000	25.000000	16.000000	16.000000
3	84.000000	21.000000	16.000000	16.000000
4	68.000000	17.000000	16.000000	16.000000
5	52.000000	13.000000	16.000000	16.000000
6	36.000000	9.000000	16.000000	16.000000
7	20.000000	5.000000	16.000000	16.000000
8	4.000000	1.000000	16.000000	16.000000
9	-12.000000	-3.000000	16.000000	4.000000
Actual Project Execution(Traditional Project)				
0	132.000000	33.000000	16.000000	00.000000
1	118.407692	29.922948	18.407692	13.592308
2	97.806768	24.159262	16.214461	20.600924
3	79.598375	19.633736	14.220528	18.208393
4	62.248551	15.144899	11.091233	17.349824
5	50.627321	12.586164	10.561235	11.621230
6	39.934610	9.966123	10.429760	10.692711
7	30.439009	7.734307	11.363918	9.495601
8	20.407625	5.279578	12.696453	10.031384
9	5.309195	1.007035	10.294475	15.098430
10	-4.588661	-1.094284	10.691094	5.309195

Table 1 Representation of Planned and Actual Traditional Project Status

7.2 Hybrid Software Project Tracking

Fig. 4 shows the deviation for Remaining _Requirements and Team_Capability individually and cumulatively.



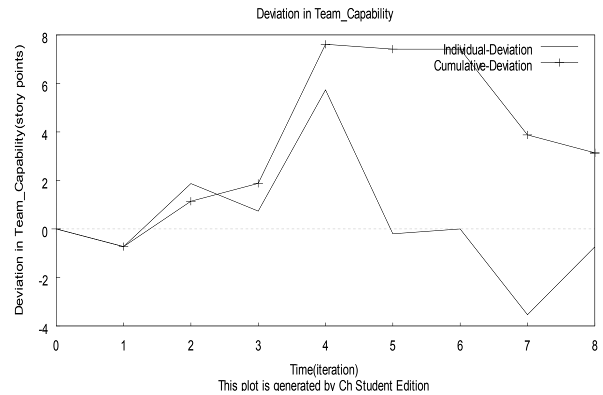
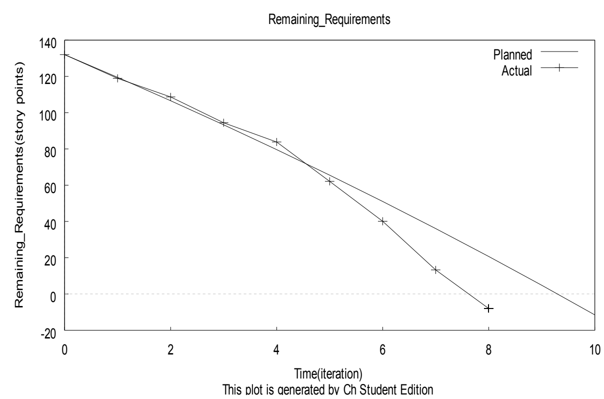
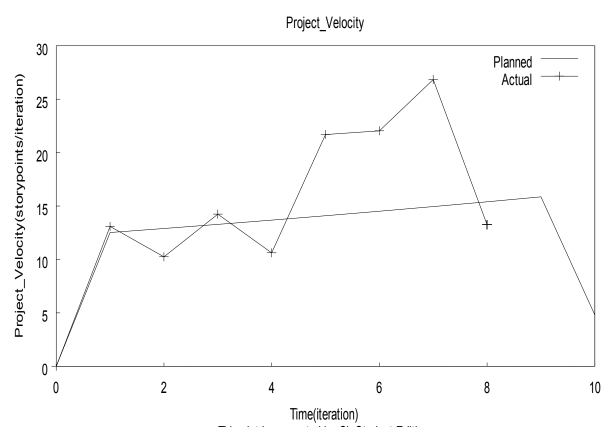
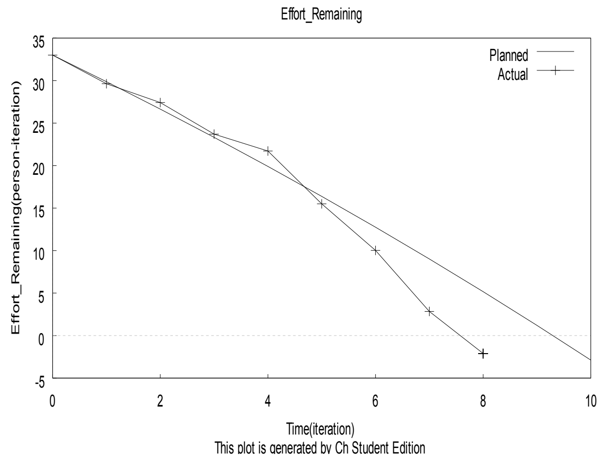


Fig. 4 Deviation for Remaining_Requirements and Team_Capability for Hybrid Project

Fig. 5 shows the project status with Effort _Remaining, Team_Strength, Remaining _ Require-ments and Project_Velocity curves. The project has completed during 8th iteration before planned duration during 10th iteration.







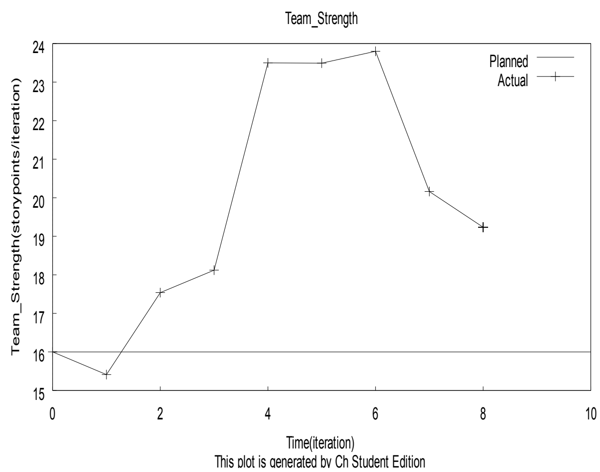


Fig. 5 Representation of Hybrid Project Status

Table 2 shows the planned and actual project status by showing Remaining_Requirements, Effort_ Remaining, Team_Strength, and Project_Velocity at each iteration.

Note that, during the last iteration, "Remaining_Requirements" and "Effort_Remaining" are negative. As

$$RR_{t+1} = -(\gamma + 1) \times P_t + (\gamma + 1) \times RR_t + \varepsilon_{ut+1} \qquad (26)$$

Depending on RR_t, P_t, and ϵ_{ut+1}; RR_{t+1} becomes negative.

$$Effort\_Remaining = \beta \times Remaining\_Requirements + \varepsilon_{\beta t} \qquad (27)$$

Effort_Remaining as well become negative. That means project actually completed somewhere during the last iteration.

Planned Project Execution (Hybrid Project)				
Iteration #	Remaining_Requirements (story points)	Effort_Remaining (person iteration)	Team_Strength (story points/iteration)	Project_Velocity (story points/iteration)
0	132.000000	33.000000	16.000000	00.000000
1	119.480000	29.870000	16.000000	12.520000
2	106.584400	26.646100	16.000000	12.895600
3	93.301932	23.325483	16.000000	13.282468
4	79.620990	19.905247	16.000000	13.680942
5	65.529620	16.382405	16.000000	14.091370
6	51.015508	12.753877	16.000000	14.514112
7	36.065973	9.016493	16.000000	14.949535
8	20.667953	5.166988	16.000000	15.398020
9	4.807991	1.201998	16.000000	15.859962
10	-11.527769	-2.881942	16.000000	4.807991
Actual Project Execution (Hybrid Project)				
Iteration #	Remaining_Requirements (story points)	Effort_Remaining (person iteration)	Team_Strength (story points/iteration)	Project_Velocity (story points/iteration)
0	132.000000	33.000000	16.000000	00.000000
1	122.478759	31.033950	19.091504	9.521241
2	103.643982	25.533865	16.158627	18.834777
3	92.226602	23.346122	18.340777	11.417380
4	75.885768	18.967448	18.117447	16.340834
5	56.084808	13.519283	14.595217	19.800960
6	39.307434	9.378056	11.062387	16.777374
7	25.665553	5.967586	7.529556	13.641881
8	21.981294	5.855198	10.932873	3.684259
9	11.727184	2.998233	11.290926	10.254110
10	3.429880	1.270461	14.363641	8.297304
11	-6.825170	-1.151226	18.937458	3.429880

Table 2 Representation of Planned and Actual Hybrid Project Status

Remarks: With simulation results for traditional/hybrid project tracking following observations are to be noted-

- The Project_Velocity is varying for hybrid project due to requirements volatility phenomena, while uniform for traditional project in plan-space.
- Remaining_Requirements decreases with iteration.
- Team-strength is uniform in plan-space and varying in execution-space.
- The project is completed before or after the planned duration as Team_Strength in execution- space, on an average, is more or less than plan- space.

8 Conclusion

Traditional/Hybrid software project tracking technique has been developed with state-space approach to track software project in plan-space and execution-space. This consists of project state transition equation and project measurement equation. The project measurement equation is used to derive project status as a function of the project state. The project state is derived with project state transition equation. The traditional/hybrid software project tracking technique is shown to be able to represent the status of software project in plan-space and execution-space at macro-level, i.e., the project status is checked at the end of each iteration by measuring the effort needed to complete the project, remaining requirements to complete the project, team strength representing the team velocity and the project progress with project velocity. The project-state is described with state-variables as requirements selected to be completed during an iteration, and remaining requirements to be completed for project completion respectively. The effect of uncertainty on project status, and the project ends during the last iteration, has shown with simulation. The ontological uncertainty has been modelled with a Normal-Distribution by using an approximation method. The ontological uncertainty has shown graphically with iteration, individually and cumulatively during project execution.

References:

[1] Roger Atkinson, Lynn Crawford, and Stephen Ward , “Fundamental Uncertainties in Projects and The Scope of Project Management,” International Journal of Project Management ,Volume 24, Issue 8, November 2006, pp. 687–698.

[2] Ahmed Al-Emran, Puneet Kapur, Dietmar Pfahl, Guenther Ruhe, “Studying the Impact of Uncertainty in Operational Release Planning – An Integrated Method and Its Initial Evaluation,” Information and Software Technology, Vol.52, 2010, pp. 446–461.

[3] Nozer D. Singpurwalla, Simon P. Wilson, “Statistical Methods in Software Engineering Reliability and Risk,” Springer Series in Statistics, May 1999.

[4] T. Dyba, “Improvisation in Small Software Organizations,” IEEE Software, Vol. 17, No. 5, 2000, pp. 82–87.

[5] S. Nerur, R. Mahapatra, G. Mangalaraj, “Challenges of Migrating to Agile Methodologies,” Communications of the ACM, May 2005, pp. 72–78.

[6] B. Boehm, “Get ready for Agile Methods, with care,” IEEE Computer, Vol. 35, No.1, 2002, pp. 64–69.

[7] Boehm, B. and Turner, R., “Balancing Agility and Discipline: A Guide for the Perplexed,” Addison-Wesley, Boston, MA, 2004.

[8] Barry W. Boehm, “Software Engineering Economics,” IEEE Transactions on Software Engineering, Volume SE-10, Number 1, January1984,pp. 4-21.

[9] Lung Chun Liu, Ellis Horowitz, “A Formal Model for Software Project Management,” IEEE Transactions on Software Engineering, Vol. 15. No. 10, October 1989, pp. 1280-1293.

[10] Olga Perminova, Magnus Gustafsson,and Kim Wikström , “Defining Uncertainty in Projects – A New Perspective,” Iinternational Journal of Project Management ,Vol.26, Issue 1, January 2008, pp. 73–79.

[11] Anders Soderholm , “ Project Management of Unexpected Events,” International Journal of Project Management, Volume 26, Issue 1, January 2008, pp. 80–86.

[12] E. Kutsch, M. Hall, “Intervening Conditions on the Management of Project Risk: Dealing with Uncertainty in Information Technology Projects,” International Journal of Project Management, Volume 23, Issue 8, November 2005, pp. 591–599.

[13] “Statistical Software Engineering”, Panel on Statistical Methods in Software Engineering, National Research Council, National Academy Press, Washington, D.C. 1996.

[14] Walker Royce, “Successful Software Management Style: Steering and Balnce,” IEEE Software Published by the IEEE Computer Society, September/ October 2005, pp. 40-47.

[15] Lee G., Murata T., “A β -Distributed Stochastic Petri Net Model for Software Development Time/Cost Management,” Journal of Systems and Software, Vol. 26, 1994,pp.149-165.

[16] Marius Vetrici, “Software Project Duration Estimation Using Metrix Model,” Revista Informatica Economicănr.3 (47)/2008, pp.87-91.

[17] Masateru Tsunoda, Tomoko Matsumura, and Ken-Itchi Matsumoto, “Modeling Software Project Monitoring with Stakeholders,” 9th IEEE/ACIS International Conference on Computer and Information Science, 2010, pp.723-728.

[18] Dimitrios Settas, Stamatia Bibi, Panagiotis Sfetsos, Ioannis Stamelos, Vassilis Gerogiannis, “Using Bayesian Belief Networks to Model Software Project Management Antipatterns,”

Dept. of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece.
[19] Bob Hughes, Mike Cotterell, Rajib Mall, "Software Project Management," Tata McGraw Hill Education Private limited, New Delhi, 5th Edition, 2011.
[20] Gabrial A. Barraza, W. Adward Back, and Fernando Mata, "Probabilistic Monitoring of Project Performance using SS- Curves," Journal of Construction Engineering and Management, ASCE, March/April 2000, pp.142-148.
[21] Roland S. Burns, "Advanced Control Engineering," Butterworth-Heinemann, 1st Edition, 2001, pp. 232.
[22] Greg Welch, and Gary Bishop, "An Introduction to the Kalman Filter," SIGGRAPH 2001, Los Angeles, CA, August 12-17, 2001, pp.15-16.
[23] Mike Cohn, "Agile Estimating and Planning," Pearson, Third Impression 2012, pp.261-312.
[24] D.J.C. Mackay, "Introduction to Monte Carlo Methods," Department of Physics, Cambridge University, Cavendish Laboratory, Madingley Road, Cambridge, CB30HE , United Kingdom.
[25] B. Brehmer, "Dynamic Decision Making: The Control of Complex Systems," Acta Psychol., Vol. 81, pp. 211–241, 1992.
[26] Peter S. Maybeck, "Stochastic Models, Estimation, and Control," Volume 1, Academic Press , Inc.(London) Ltd,pp.1-3.
[27] Steiner, I.D., "Models for inferring Relationships between Group size and Potential Group Productivity," Behavioral Science, 1966, pp.273-283.
[28] Claudia de O. Melo, Daniela S. Cruzes, Fabio Kon, and , Reidar Conradi, " Interpretative Case Studies on Agile Team Productivity and Management," Information and Software Technology , Vol.55, 2013,pp. 412–427.
[29] B Lakhanpal, "Understanding the Factors influencing the Performance of Software Development Groups: An Exploratory Group-Level Analysis," Information and Software Technology, Volume 35, Issue 8, August 1993, pp. 468–473.
[30] D. Rodriguez, M.A. Sicilia, E. Garcíaa, and R. Harrison, "Empirical Findings on Team Size and Productivity in Software Development," The Journal of Systems and Software ,Vol. 85,March 2012,pp. 562– 570.
[31] Christof Ebart, Jozef De Man, "Requirements Uncertainty: Influencing Factors and Concrete Improvements," ICSE'05, St. Louis , Missouri,USA, May 15-21,2005, pp. 553-560.
[32] D. Pfahl, and K. Lebsanft "Using Simulation to Analyse the Impact of Software Requirement Volatility on Project Performance," Information and Software Technology, Vol. 42, 2000, pp. 1001-1008.
[33] Tony Moynihan, "Coping with Requirements-Uncertainty: The Theories-of-Action of Experienced IS/Software Project Managers," The Journal of Systems and Software, Vol. 53,August 2000,pp. 99-109.
[34] Susan Ferreira, James Collofello, Dan Shunk, and Gerald Mackulak, " Understanding the Effects of Requirements Volatility in Software Engineering by using Analytical Modeling and Software Process Simulation," The Journal of Systems and Software ,Vol. 82,October 2009,pp. 1568–1577.
[35] kim Bang Salling, " Risk Analysis and Monte Carlo Simulation within Transport Appraisal," Centre for Traffic and Transport, CTT-DTU, Build. 115, Technical University of Denmark DK- 2800 Lyngby, Denmark, pp.3-5.
[36] Martin Hilbert, "Toward a Synthesis of Cognitive Biases: How Noisy Information Processing can bias Human Decision Making," Psychological Bulletin, Vol.138, Issue 2, March 2012, pp. 211-237.
[37] Geoffrey Gordon, "System Simulation", Second Edition, Pearson -Prentice Hall, pp.40-41, 158.