CSIT (June 2014) 2(2):141–151
DOI 10.1007/s40012-014-0037-5

Traditional and hybrid software project tracking technique formulation: state space approach with initial state uncertainty

Manoj Kumar Tyagi · Srinivasan Munisamy · L. S. S. Reddy

Received: 27 November 2013 / Accepted: 8 March 2014 / Published online: 9 September 2014

Abstract Software projects are required to be tracked during their execution for controlling them. According to state-space approach, the tracking technique consists of software project state transition equation and software project status measurement equation. A key factor in tracking software projects is to represent the project with uncertainty involved in the parameters. Traditional and hybrid software project tracking technique is designed with state space approach and simulated using discrete event simulation in plan-space and execution-space. The uncertainty considered here is epistemological and is modeled as a normal distribution using an approximation method. The initial state of the project in execution-space also has an uncertainty associated with it. The project tracking technique consists of project state transition equation and project status measurement equation, in plan-space and is formulated with Monte Carlo method in execution-space. The project status is derived using project measurement equation as a function of project state. The project state is derived using project state transition equation. The software product is developed iteratively and incrementally. With Monte Carlo simulation runs, simulation result shows the uncertainty propagation iteration-wise, both individually and totally; and the effect of uncertainty on project status is shown by showing project status in execution-space and plan-space. Besides, the project completion somewhere during the last iteration is shown with simulation.

Keywords Epistemological uncertainty · Normal distribution function · State-space approach · Hybrid project · Traditional project · Monte Carlo simulation · Discrete event simulation · Execution-space · Plan-space

M. K. Tyagi (✉)
Electronics and Computer Engineering, K L University, Vaddeswaram 522502, AP, India
e-mail: manojkumar@kluniversity.in

S. Munisamy
Electronics and Communication Engineering, Meerut Institute of Engineering and Technology, Meerut 250005, UP, India
e-mail: msrinivasan77@yahoo.com

L. S. S. Reddy
K L University, Vaddeswaram 522502, AP, India
e-mail: drlssreddy@kluniversity.in

1 Introduction

How software projects are modeled for tracking by considering state-space view? Software project tracking is defined as determining the project status during its execution and comparing it with the project status during its planning. Project tracking is needed to control software projects.

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. The plan-space and execution-space consists of project states through which software projects transit. 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 ([27], pp. 2–4).

We have considered the unknown factors: team productivity and ambiguity about the product requirements (features). Team productivity [25] refers to number of requirements completed during an iteration. Ambiguity about the product requirements (features) is represented with requirements volatility phenomena. Requirements volatility [13, 14, 22, 24] refers to growth or changes in requirements during a project's development lifecycle. Requirements volatility represents three types of requirements changes requested by customers: new requirements to be added, requirements to be deleted and requirements generated due to modifications to specified requirements.

Software product development is based on traditional, agile, and hybrid software development philosophy [9, 12]. Both agile and traditional philosophies have a home ground area of project characteristics in which each clearly works best, and where the other will have difficulties [5]. Hybrid philosophy that combines both philosophies is required for projects, which includes agile and plan-driven home ground characteristics [5, 6]. In this paper, hybrid philosophy is developed with considering requirements volatility phenomena partially, i.e., accepting the modifications to the already specified requirements which were documented before starting the project execution, but not accepting the requests for requirements changes (new requirements to be added, requirements to be deleted) by customers during execution. Traditional philosophy does not consider requirements volatility phenomena at all.

There has been considerable research 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 [4, 20], as opposed to model the software projects with uncertainty [17–23] 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 [29]. In execution space, software projects are usually started with absolutely determined project state which is identical to the state in plan space. In this paper, traditional/hybrid software project tracking technique is designed with state-space approach considering the uncertainty associated with project initial state, team productivity, and requirement's-specification. The software product is developed iteratively and incrementally [26].

2 Related work

Software project modeling has been an interesting area for researchers since 1950 s onwards. Traditional-models such as Work Breakdown Structure, Gantt Model, Critical Path Method (CPM) Model, PERT Model, AND-OR Graph Model, Petri Net Model, Stochastic Petri net-based Model, Design Net Model [19, 20] 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 [32]. 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, key performance indicator, corrective action, and check timing [31]. 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 [35]. The software project tracking literature [3, 16] 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. The above specified tracking techniques are used for cost or schedule tracking.

Having gone through the literature, we have 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 tracking.
- Little consideration has been given to project modeling for tracking with due consideration for uncertainty related to software development productivity, requirement's-specification document.
- Uncertainty associated with project initial state has not been considered by existing tracking techniques.

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

3 Design decisions

The traditional/hybrid software project tracking technique is developed using the state space approach [8, 34]. The state space model is developed based on a case study described in Cohn's book [10]. The case study here involves a mythical game-development company "Bomb Shelter Studios". In this study, a game termed 'Havannah' was developed iteratively and incrementally using agile philosophy. But we have considered to develop the product with traditional and hybrid philosophy. The important design decisions are related to the dynamics of product evolution, dynamics of requirements generation, and uncertainty representation.

3.1 Dynamics of product evolution

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

3.2 Dynamics of requirements generation

At the start of every iteration, a fixed set of frozen-requirements (those requirements which once selected for increment, are never changed during product development) are selected from requirement's specification document to start with. Traditional projects are neutral to requirements volatility phenomena, i.e., requirements changes are not entertained. Hybrid projects respond partially to requirements volatility phenomena, i.e., requirements generated due to modifications requested by customers to leftover requirements are entertained. These new requirements do not produce major changes in the software being built, but leads to more customer satisfaction. We assume that the new requirements do not reflect modification or replacement of already implemented requirements.

3.3 Uncertainty representation

Uncertainty has been defined in [33] as any deviation from the unachievable ideal of completely deterministic knowledge of the relevant system. We consider epistemological uncertainty defined by [33] as the imperfection of our knowledge about a fact that cannot be guaranteed to be consistent or correct (i.e., most likely to be overestimated or underestimated). Epistemological uncertainty factors considered here are: team productivity, requirements generated due to modifications to specified requirements (requirements volatility).

Epistemological uncertainty 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 ([15], p. 158)

$$y = \frac{\sum_{i=1}^{k} x_i - \frac{k}{2}}{\left(\frac{k}{12}\right)^{1/2}} \quad \text{as} \quad k \to \infty \equiv (0{:}1), \tag{1}$$

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

4 Design of traditional/hybrid software project tracking technique

State-Space approach [8, 34] is used to design software project tracking technique in plan-space, and as well using Monte Carlo method in execution-space. The Monte Carlo method [15], pp. 40–41) refers a numerical computation technique which consists of experimental sampling of random numbers.

The software project tracking technique consists of software project state transition equation and software project status measurement equation. The software project state transition equation is used to derive the project state which is used by software project measurement equation to determine project status. The following equations (A) and (B) are used to represent a software project tracking technique in plan-space and execution-space, respectively.

Software project state transition equation

$$\vec{S}_{t+1} = A\vec{S}_t.$$

Software project status measurement equation

$$\vec{M}_t = H\vec{S}_t. \tag{A}$$

Software project state transition equation

$$\vec{S}_{t+1} = A\vec{S}_t + \vec{N}_t + 1.$$

Software project status measurement equation

$$\vec{M}_t = H\vec{S}_t, \tag{B}$$

where A and H represent a state-transition matrix and measurement transition matrix, respectively. $\vec{N}$ represents a project state noise (uncertainty) 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

Software projects execute in plan-space and execution-space. The software project, during execution, is described as a stochastic-process [7, 21]; which changes its state over the life of a project. The traditional/hybrid software project tracking technique is developed using state-space approach

Fig. 1 Representation of project-state

in plan-space and as well using Monte Carlo method in execution-space.

5.1 Traditional and hybrid software project tracking technique formulation

Software project deviates from plan during its execution. This happens due to uncertainty factors: team productivity and requirements volatility. At time 't', the project state is represented with state vector $[P_t, RR_t]^T$, where T represents the matrix transpose operation.

P_t: team-capability representing team productivity during $(t + 1)$th iteration.

RR_t: remaining-requirements to be completed for project completion at the end of tth iteration.

P_t can be estimated with three techniques—Historical Values, Make a Forecast, and Run an Iteration [10]. Here, it was estimated initially with historical values technique.

The project is started with initial-state $[P_0, RR_0]^T = [16, 132]^T$ in plan-space and initial-state $[P_0 + \varepsilon_{p0}, RR_0]^T = [16 + \varepsilon_{p0}, 132]^T$ in execution space, where ε_{p0} represents uncertainty associated with project initial state.

Figure 1 represents the project state transition during $(t + 1)$th iteration.

The team-capability 'P_t' is uniform in plan-space and varying with epistemological uncertainty in execution-space. It is affected with the factors such as schedule-pressure, communication/motivation, size-estimation, workforce-experience level [11, 18, 25, 30] etc. in execution-space. Their net-effect on team-capability is a random variable "ε_p" representing epistemological uncertainty. Mathematically,

$$P_{t+1} = P_t \quad \text{plan-space}, \tag{2}$$

$$P_{t+1} = P_t + \epsilon_{pt+1} \quad \text{execution-space}, \tag{3}$$

where ε_{pt+1} represents epistemological uncertainty introduced during $(t + 1)$th iteration associated with team-capability.

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 requirements 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 is calculated as $(RR_t - P_t)$.

(iii) Requirements generated due to modifications in left over requirements $(RR_t - P_t)$ is calculated as $\gamma(RR_t - P_t)$, where "γ" is a stochastic variable controlling the number of requirements generated due to modifications to left over requirements. "γ" is uniform in plan-space and introduces the epistemological uncertainty, represented by "ε_u", in RR_{t+1}, in execution-space.

(iv) Remaining-requirements at the end of $(t + 1)$th iteration is calculated by adding: $(RR_t - P_t)$, $\gamma(RR_t - P_t)$ in plan-space and in addition to it, adding ε_{ut+1}, in execution-space.

Mathematically,

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

$$RR_{t+1} = -(\gamma + 1)P_t + (\gamma + 1)RR_t \quad \text{plan-space}, \tag{4}$$

$$RR_{t+1} = -(\gamma + 1)P_t + (\gamma + 1)RR_t + \epsilon_{ut+1} \quad \text{execution-space}, \tag{5}$$

where ε_{ut+1} represents epistemological uncertainty introduced during $(t + 1)$th iteration associated with remaining-requirements.

The uncertainties "ε_p" and "ε_u" are modeled using Eq (1).

The software project state transition equation is developed by using difference equations describing the dynamics of software project in plan-space. Equations (2) and (4) describe the software project state transition in plan-space.

$$P_{t+1} = P_t,$$

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

$$P_{t+1} = 1 \times P_t + 0 \times RR_t, \tag{6}$$

$$RR_{t+1} = -(\gamma + 1) \times P_t + (\gamma + 1) \times RR_t. \tag{7}$$

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:

$$\text{Remaining requirements} = RR_t,$$

$$\text{Project velocity} = RR_{t-1} - RR_t,$$

$$\text{Effort remaining} = \beta \times RR_t,$$

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 $ 2,000/person-iteration, the cost/story-points is 2,000/4 = 500 $/story-points. Then β is calculated as follows

$$\beta = (500\ \$/\text{story-points})/(2{,}000\ \$/\text{person-iteration}) = 1/4\ \text{person-iteration/story-points},$$

$$\text{Team strength} = P_t,$$

$$\text{Remaining requirements} = 0 \times P_t + 1 \times RR_t, \quad (8)$$

$$\text{Project velocity} = RR_{t-1} - RR_t, \quad (9)$$

$$\text{Effort remaining} = 0 \times P_t + \beta \times RR_t, \quad (10)$$

$$\text{Team strength} = 1 \times P_t + 0 \times RR_t. \quad (11)$$

The software project state transition equation in matrix form in plan-space is derived using Eqs. (6), and (7); and software project measurement equation in matrix form in plan-space is derived using Eqs. (8), (10) and (11). Equation (12) represents a state space model for software project 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 status measurement equation (12)

$$\begin{bmatrix} \text{Remaining requirements} \\ \text{Effort remaining} \\ \text{Team strength} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \beta \\ 1 & 0 \end{bmatrix} \begin{bmatrix} P_t \\ RR_t \end{bmatrix}.$$

Equations (9) and (12) are used to track the status of software project in plan-space.

The software project state transition equation is developed by using difference equations describing the dynamics of software project in execution-space. Equations (3) and (5) describe the software project state transition in execution-space.

$$P_{t+1} = P_t + \epsilon_{pt+1},$$

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

$$P_{t+1} = 1 \times P_t + 0 \times RR_t + \epsilon_{pt+1}, \quad (13)$$

$$RR_{t+1} = -(\gamma+1) \times P_t + (\gamma+1) \times RR_t + \epsilon_{ut+1}. \quad (14)$$

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:

$$\text{Remaining requirements} = RR_t,$$

$$\text{Project velocity} = RR_{t-1} - RR_t,$$

$$\text{Effort remaining} = \beta \times RR_t,$$

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 \$ 2,000/person-iteration, the cost/story-points is 2,000/4 = 500 \$/story-points. Then β is calculated as follows

$$\beta = (500\ \$/\text{story-points})/(2{,}000\ \$/\text{person-iteration})$$
$$= 1/4\ \text{person-iteration/story-points},$$

$$\text{Team strength} = P_t,$$

$$\text{Remaining requirements} = 0 \times P_t + 1 \times RR_t, \quad (15)$$

$$\text{Project velocity} = RR_{t-1} - RR_t, \quad (16)$$

$$\text{Effort remaining} = 0 \times P_t + \beta \times RR_t, \quad (17)$$

$$\text{Team strength} = 1 \times P_t + 0 \times RR_t. \quad (18)$$

The software project state transition equation in matrix form in execution-space is derived using Eqs. (13), and (14); and software project measurement equation in matrix form in execution-space is derived using Eqs. (15), (17), and (18). Equation (19) represents a state space model for software project in execution-space.

Software project state transition equation

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

Software project status measurement equation

$$\begin{bmatrix} \text{Remaining requirements} \\ \text{Effort remaining} \\ \text{Team strength} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & \beta \\ 1 & 0 \end{bmatrix} \begin{bmatrix} P_t \\ RR_t \end{bmatrix}. \quad (19)$$

Equations (16) and (19) are used to track the status of software project in execution-space.

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

- If γ and ε_u are zero for all iterations then the model represents a traditional project tracking technique.
- If γ 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 and remaining requirements used for modeling. The uncertainty propagation is described with iteration, individually and totally, in execution-space.

At time t = 0, i.e., during first iteration

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

$RR = -(\gamma + 1)P_0 + (\gamma + 1)RR_0 + \epsilon_{u1}$.

Clearly, uncertainty with team capability during first iteration,

$\text{Individual-Deviation\_TC}_1 = \epsilon_{p1}$,

$\text{Total-Deviation\_TC}_1 = \epsilon_{p1}$.

Uncertainty with remaining requirements during first iteration,

$\text{Individual-Deviation\_RR}_1 = \epsilon_{u1}$,

$\text{Total-Deviation\_RR}_1 = \text{Individual-Deviation\_RR}_1$.

At time t = 1, i.e., during second iteration

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

$RR_2 = (\gamma + 1)P_1 + (\gamma + 1)RR_1 + \epsilon_{u2}$.

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

$\text{Individual-Deviation\_TC}_2 = \epsilon_{p2}$,

$\text{Total-Deviation\_TC}_2 = \epsilon_{p1} + \epsilon_{p2}$.

Uncertainty with remaining requirements during second iteration,

$\text{Individual-Deviation\_RR}_2 = \epsilon_{u2}$,

$$\begin{aligned}\text{Total-Deviation\_RR}_2 &= -(\gamma + 1)\epsilon_{p1} + (\gamma + 1)\epsilon_{u1} + \epsilon_{u2} \\ &= -(\gamma + 1)\text{Total-Deviation\_TC}_1 \\ &\quad + (\gamma + 1)\text{Total-Deviation\_RR}_1 \\ &\quad + \text{Individual-Deviation\_RR}_2.\end{aligned}$$

By generalizing, during (t + 1)th iteration, uncertainty with team capability, individually and totally

$$\text{Individual-Deviation\_TC}_{t+1} = \epsilon_{pt+1}, \tag{20}$$

$$\text{Total-Deviation\_TC}_{t+1} = \epsilon_{p1} + \epsilon_{p2} + \cdots + \epsilon_{pt+1}. \tag{21}$$

Uncertainty with remaining requirements, individually and cumulatively

$$\text{Individual-Deviation\_RR}_{t+1} = \epsilon_{ut+1}, \tag{22}$$

$$\begin{aligned}\text{Total-Deviation\_RR}_{t+1} &= -(\gamma + 1)\text{Total-Deviation\_TC}_t \\ &\quad + (\gamma + 1)\text{Total-Deviation\_RR}_t \\ &\quad + \text{Individual-Deviation\_RR}_{t+1}.\end{aligned} \tag{23}$$

7 Simulation results

The simulation is performed using Ch Interpreter (C/C ++ Interpreter) developed by Dr. Harry H. Chang [37]. The project status is tracked at the end or starting of iteration.

7.1 Traditional software project tracking

The project is initialized with the state vector $[16, 132]^T$ in plan space and $[16 + \varepsilon_{p0}, 132]^T$ in execution space. 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 totally, respectively; which affects the status of the project in execution-space. Figure 2 shows the deviation for Remaining_Requirements and Team_Capability individually and totally. The Individual-Deviation curve for deviation in 'Remaining_Requirements' is zero, as 'γ' is zero. But, the Total-Deviation curve for deviation in 'Remaining_Requirements' at the end of an iteration, is varying due to 'TotalDeviation_TC' related to 'Team_Capability' and 'TotalDeviation_RR' related to 'Remaining_Requirements' with reference to previous iteration, as evident with Eq (23).

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. Figure 3 shows the project behavior with Effort_Remaining, Team_Strength, Remaining_Requirements and Project_Velocity curves. The project status curves are used by project managers to assess whether a project is progressing according to plan or not, with using project status variable. For example, at the end of fourth iteration, project work ('Remaining_Requirements') is completing faster than plan, effort required to complete the remaining work ('Effort_Remaining') is lesser than plan, 'Team_Strength' is greater than plan, and project velocity ('Project_Velocity') is greater than plan, as shown in 'Fig. 3' and 'Table 1'. The project is completed during seventh iteration before planned duration during ninth iteration.

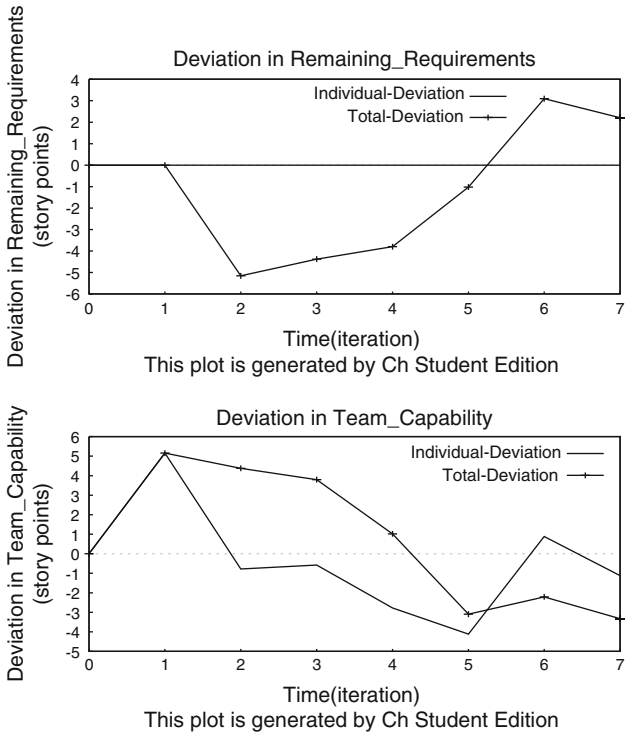

Fig. 2 Deviation for Remaining_Requirements and Team_Capability for traditional project

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

7.2 Hybrid software project tracking

The project is initialized with the state vector $[16, 132]^T$ in plan-space and $[16 + \varepsilon_{p0}, 132]^T$ in execution-space. The software project changes its state because of the requirements completed by the developers, new requirements generated due to customer requests for modifications in leftover requirements during an iteration. 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.

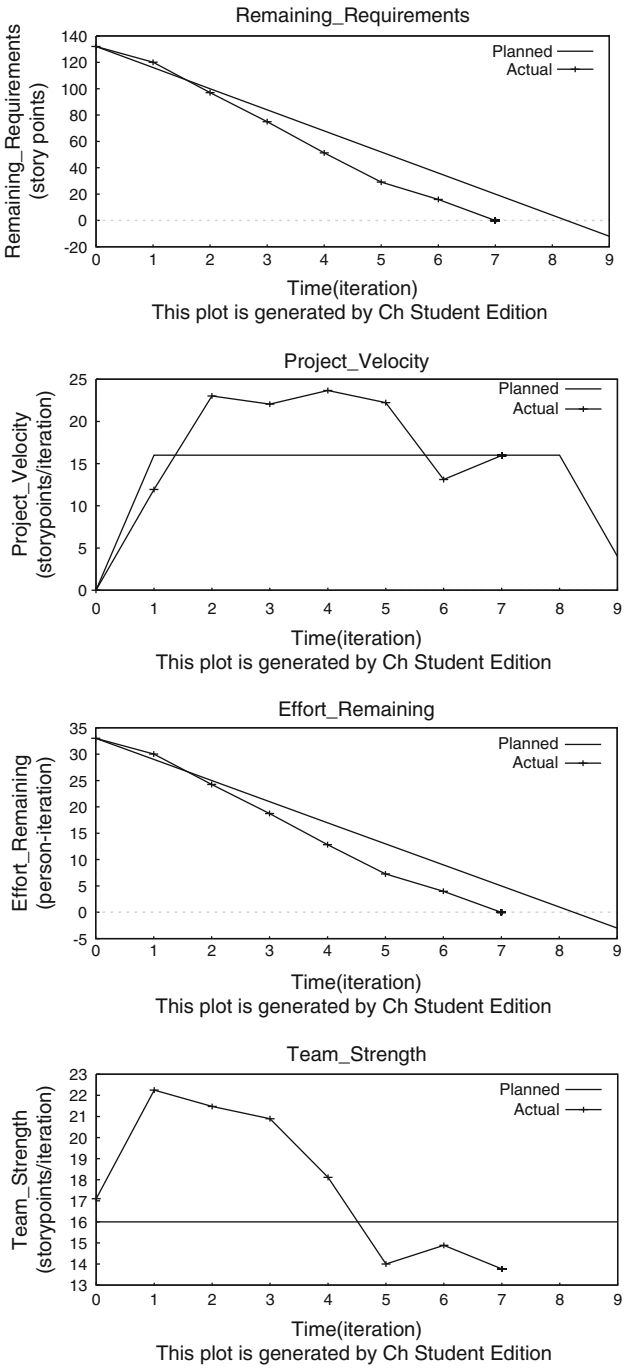

Fig. 3 Representation of traditional project status

With Monte Carlo runs, the deviation for Remaining_Requirements and Team_Capability has been shown iteration-wise, individually and totally, respectively; which affects the status of the project in execution-space. Figure 4 shows the deviation for Remaining_Requirements and Team_Capability individually and totally. The Individual-Deviation curve for deviation in 'Remaining_Requirements'

Table 1 Representation of planned and actual project status

Iteration numbers	Remaining_Requirements (Story points)	Effort_Remaining (Person iteration)	Team_Strength (Story points/ iteration)	Project_Velocity (Story points/ iteration)
Planned project execution (traditional project)				
0	132.000000	33.000000	16.000000	0.000000
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	17.093540	0.000000
1	120.063379	30.015845	22.250460	11.936621
2	97.034478	24.258620	21.472018	23.028901
3	74.983315	18.745829	20.892873	22.051163
4	51.310450	12.827613	18.112881	23.672864
5	29.083063	7.270766	13.998375	22.227387
6	15.969626	3.992407	14.883314	13.113437
7	−0.030447	−0.007612	13.766555	15.969626

is varying, as 'γ' is non-zero. But, the Total-Deviation curve for deviation in 'Remaining_Requirements' at the end of an iteration, is varying due to 'TotalDeviation_TC' related to 'Team_Capability' and 'TotalDeviation_RR' related to 'Remaining_Requirements' with reference to previous iteration, and 'IndividualDeviation_RR' related to 'Remaining_Requirements' as evident with Eq (23).

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. Figure 5 shows the project behavior with Effort_Remaining, Team_Strength, Remaining_Requirements and Project_Velocity curves. The project status curves are used by project managers to assess whether a project is progressing according to plan or not, with using project status variables. For example, at the end of fourth iteration, project work ('Remaining_Requirements') is completing slower than plan, effort required to complete the remaining work ('Effort_Remaining') is more than plan, 'Team_Strength' is lesser than plan, and project velocity ('Project_Velocity') is lesser than plan, as shown in 'Fig. 4' and 'Table 2'. The project is completed during 10th iteration after planned duration during 10th iteration.

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

$$RR_{t+1} = -(\gamma + 1)P_t + (\gamma + 1)RR_t + \epsilon_{ut+1}. \quad (24)$$

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

$$\text{Effort\_Remaining} = \beta \times \text{Remaining\_Requirements}. \quad (25)$$

Effort-Remaining as well become negative. That means project actually completed somewhere during the last iteration.

7.2.1 *Remarks*

With simulation results for traditional/hybrid project tracking, following observations are to be noted.

- At the beginning of project, the initial value of status variable "Team_Strength" in execution-space differs from that in plan-space due to uncertainty introduced in the initial state of the project in execution-space.

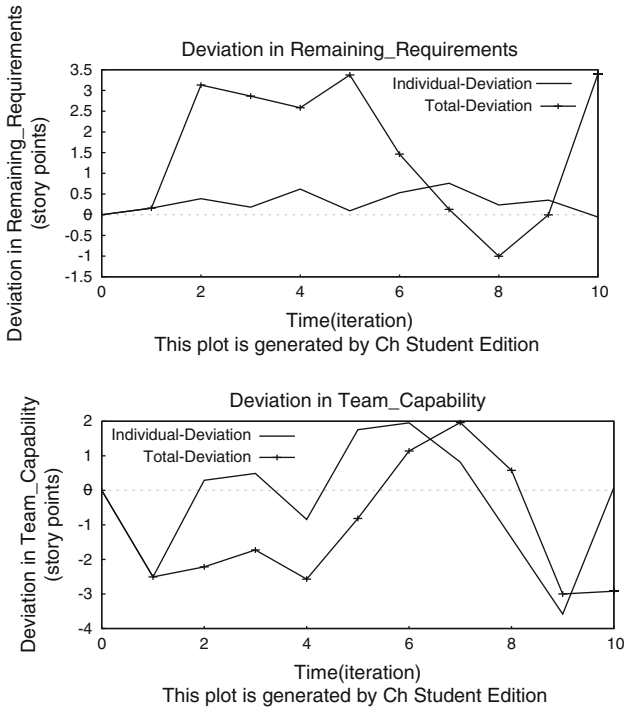

Fig. 4 Deviation for Remaining_Requirements and Team_Capability for hybrid project

- The Project_Velocity is varying in plan-space for hybrid project due to requirements volatility phenomena, while uniform for traditional project.
- Remaining_Requirements decreases with iteration.
- Effort_Remaining is in phase with Remaining_Requirements for the project.
- 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 and future work

Traditional/hybrid software project tracking technique has been developed with state-space approach 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 function of the project state. The project state is derived using project state transition equation. The software project tracking technique is able to represent the status of traditional and hybrid software project in plan-space and execution-space at macro-level, i.e., the project status is checked at the end of each iteration. The effect of uncertainty propagation on project status and the project completion during the last iteration has been shown with simulation. The epistemological uncertainty has been modeled with a normal-distribution by using an approximation method. The developed project tracking technique is to be enhanced to design an intelligent project tracking technique, which determines project status with considering uncertainty as a future work.

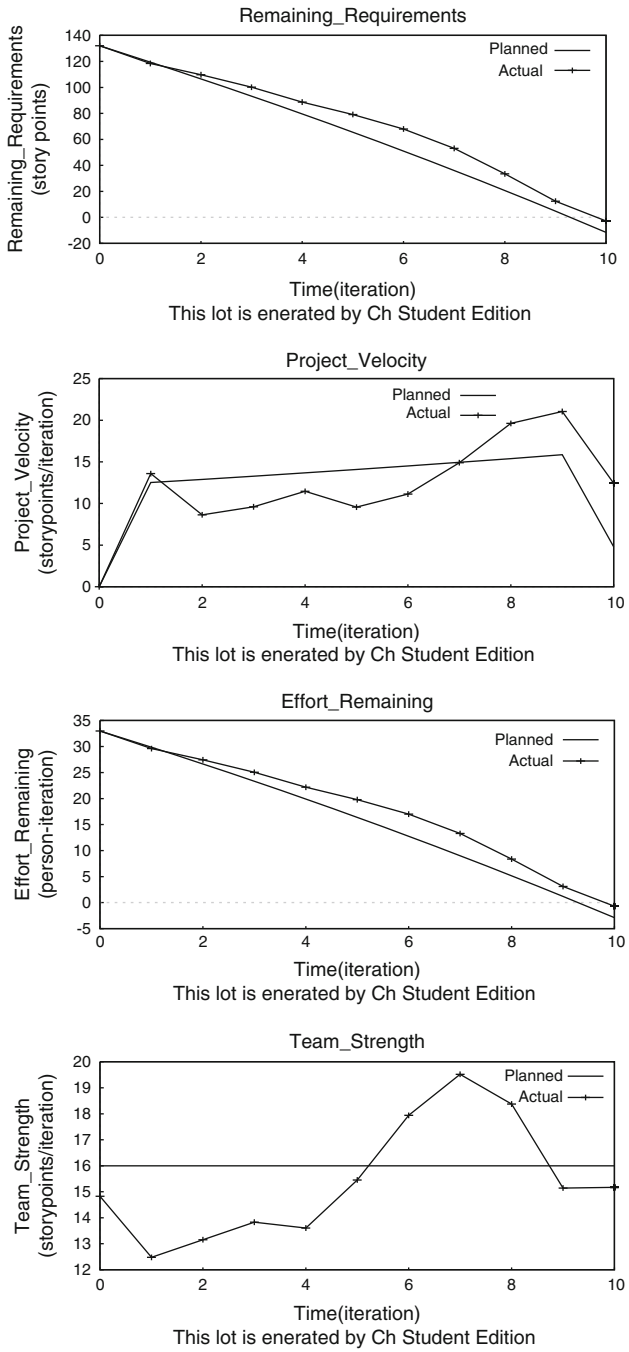

Fig. 5 Representation of hybrid project status

Table 2 Representation of planned and actual project status

Iteration numbers	Remaining_Requirements (Story points)	Effort_Remaining (Person iteration)	Team_Strength (Story points/ iteration)	Project_Velocity (Story points/ iteration)
Planned project execution (hybrid project)				
0	132.000000	33.000000	16.000000	0.000000
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)				
0	132.000000	33.000000	14.827577	0.000000
1	118.407558	29.601889	12.477023	13.592442
2	109.767179	27.441795	13.156124	8.640379
3	100.163000	25.040750	13.829952	9.604179
4	88.704639	22.176160	13.604796	11.458361
5	79.147138	19.786784	15.454589	9.557501
6	68.013205	17.003301	17.939002	11.133933
7	53.109269	13.277317	19.519248	14.903936
8	33.486967	8.371742	18.374142	19.622302
9	12.432634	3.108159	15.143651	21.054333
10	−2.767646	−0.691912	15.169115	12.432634

References

1. Al-Emran A, Kapur P, Pfahl D, Ruhe G (2010) Studying the impact of uncertainty in operational release planning—an integrated method and its initial evaluation. Inf Softw Technol 52:446–461
2. Atkinson R, Crawford L, Ward S (2006) Fundamental uncertainties in projects and the scope of project management. Int J Project Manag 24(8):687–698
3. Barraza GA, Back WA, Mata F (2000) Probabilistic monitoring of project performance using SS-curves. J Constr Eng Manag ASCE, March/April, 142–148
4. Boehm BW (1984) Software engineering economics. IEEE Trans Softw Eng SE-10(1):4–21
5. Boehm B (2002) Get ready for Agile Methods, with care. IEEE Comput 35(1):64–69
6. Boehm B, Turner R (2004) Balancing agility and discipline: a guide for the perplexed. Addison-Wesley, Boston
7. Brehmer B (1992) Dynamic decision making: the control of complex systems. Acta Psychol 81:211–241
8. Burns RS (2001) Advanced control engineering, 1st edn. Butterworth-Heinemann, Boston, p 232
9. Cockburn A (2002) Agile software development. Addison-Wesley, Boston
10. Cohn M (2012) Agile estimating and planning. Pearson, Third Impression 2012, pp 261–312
11. de O. Melo C, Cruzes DS, Kon F, Conradi R (2013) Interpretative case studies on agile team productivity and management. Inf Softw Technol 55:412–427
12. Dyba T (2000) Improvisation in small software organizations. IEEE Softw 17(5):82–87
13. Ebart C, De Man J (2005) Requirements uncertainty: influencing factors and concrete improvements. In: ICSE'05, St. Louis, Missouri, USA, 15–21 May 2005, pp 553–560
14. Ferreira S, Collofello J, Shunk D, Mackulak G (2009) Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. J Syst Softw 82:1568–1577
15. Gordon G (2008) System simulation, vol 158, 2nd edn. Pearson-Prentice Hall, Upper Saddle River, pp 40–41
16. Hughes B, Cotterell M, Mall R (2011) Software project management, 5th edn. Tata McGraw Hill Education Private Limited, New Delhi
17. Kutsch E, Hall M (2005) Intervening conditions on the management of project risk: dealing with uncertainty in information technology projects. Int J Project Manag 23(8):591–599
18. Lakhanpal B (1993) Understanding the factors influencing the performance of software development groups: an exploratory group-level analysis. Inf Softw Technol 35(8):468–473
19. Lee G, Murata T (1994) A β-distributed stochastic petri net model for software development time/cost management. J Syst Softw 26:149–165
20. Liu LC, Horowitz E (1989) A formal model for software project management. IEEE Trans Softw Eng 15(10):1280–1293
21. Maybeck PS (1979) Stochastic models, estimation, and control, vol 1. Academic Press, Inc., Ltd., London, pp 1–3

22. Moynihan T (2000) Coping with requirements-uncertainty: the theories-of-action of experienced IS/software project managers. J Syst Softw 53:99–109
23. Perminova O, Gustafsson M, Wikström K (2008) Defining uncertainty in projects—a new perspective. Int J Project Manag 26(1):73–79
24. Pfahl D, Lebsanft K (2000) Using simulation to analyse the impact of software requirement volatility on project performance. Inf Softw Technol 42:1001–1008
25. Rodriguez D, Sicilia MA, Garcíaa E, Harrison R (2012) Empirical findings on team size and productivity in software development. J Syst Softw 85:562–570
26. Royce W (2005) Successful software management style: steering and balance. IEEE Software published by the IEEE Computer Society, September/October, pp 40–47
27. Singpurwalla ND, Wilson SP (May 1999) Statistical methods in software engineering reliability and risk. Springer series in statistics, Berlin
28. Soderholm A (2008) Project management of unexpected events. Int J Project Manag 26(1):80–86
29. Statistical software engineering (1996). Panel on Statistical Methods in Software Engineering, National Research Council, National Academy Press, Washington, DC
30. Steiner ID (1966) Models for inferring relationships between group size and potential group productivity. Behav Sci 11:273–283
31. Tsunoda M, Matsumura T, Matsumoto K-I (2010) Modeling software project monitoring with stakeholders. In: 9th IEEE/ACIS international conference on computer and information science, 2010, pp 723–728
32. Vetrici M (2008) Software project duration estimation using metrix model. Rev Inform Econ 3(47):87–91
33. Walker WE et al (2003) Defining uncertainty—a conceptual basis for uncertainty management in model-based decision support. J Integr Assess 4(1):5–17
34. Welch G, Bishop G (2001) An introduction to the Kalman filter. In: SIGGRAPH 2001, Los Angeles, CA, 12–17 August 2001, pp 15–16
35. Settas D, Bibi S, Sfetsos P, Stamelos I, Gerogiannis V (2006) Using Bayesian belief networks to model software project management antipatterns. Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki
36. Salling KB (2007) Risk analysis and Monte Carlo simulation within transport appraisal. Centre for Traffic and Transport, CTT-DTU, Technical University of Denmark, Lyngby, pp 3–5
37. Ch Interpreter (Ch Student Edition V6.1) (2008) SoftIntegration, Inc.