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
\(^3\)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 stakeholders (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.

\[
\begin{align*}
\text{Software Project State Transition Model} \\
\vec{S}_{t+1} &= A \times \vec{S}_t \\
\text{Software Project Measurement Model} \\
\vec{M}_t &= H \times \vec{S}_t \\
\end{align*}
\]

\[
\begin{align*}
\text{Software Project State Transition Model} \\
\vec{S}_{t+1} &= A \times \vec{S}_t + \vec{N}_{t+1} \\
\text{Software Project Measurement Model} \\
\vec{M}_t &= H \times \vec{S}_t + \vec{V}_t \\
\end{align*}
\]

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.

\[
\begin{pmatrix}
P_t \\
RR_t
\end{pmatrix}
\quad \text{Plan Space}
\]

\[
\begin{pmatrix}
P_{t+1} \\
RR_{t+1}
\end{pmatrix}
\quad \text{Execution Space}
\]

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 \(P_t\), 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 with requirements-volatility phenomena, is \((RR_t-P_t)\); in plan-space and in addition to it adding \(\epsilon_u\); in execution-space.

\[
RR_{t+1} = Y(RR_t-P_t) + \epsilon_u
\]

Therefore, remaining-requirements at the end of \((t+1)^{th}\) iteration is found 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 \(Y(RR_t-P_t)\); in plan-space and in addition to it adding \(\epsilon_u\); in execution-space.

\[
RR_{t+1} = -(\gamma+1)\times P_t + (\gamma+1)\times RR_t
\]

\[
RR_{t+1} = -(\gamma+1)\times P_t + (\gamma+1)\times RR_t + \epsilon_u
\]

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 \(\epsilon_u\), 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 \([37], pp.158\)

\[
y = \frac{\sum_{i=1}^{k} x_i - \frac{k}{2}}{\left(\frac{k}{12}\right)^{\frac{1}{2}}} \quad \text{As } k \to \infty = (0:1)
\]

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.

\[
P_{t+1} = P_t
\]

\[
RR_{t+1} = - (Y+1)\times P_t + (Y+1)\times RR_t
\]

\[
P_{t+1} = 1 \times P_t + 0 \times RR_t
\]

\[
RR_{t+1} = - (Y+1)\times P_t + (Y+1)\times RR_t
\]

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:

\[
\text{remainingrequirements} = RR_t
\]

\[
\text{projectvelocity} = RR_{t-1} - RR_t
\]
effortremaining = $ \beta \times RR_t$

Where $\beta$ 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 $2000/4 = 500$ S/ story-points. Then $\beta$ is calculated as follows

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

teamstrength = $P_t$

remaining requirements $= 0 \times P_t + 1 \times RR_t$  \hspace{1cm} (10)

project velocity $= RR_{t+1} - RR_t$ \hspace{1cm} (11)

effortremaining $= 0 \times P_t + $ $\beta \times RR_t$  \hspace{1cm} (12)

teamstrength $= 1 \times P_t + 0 \times RR_t$ \hspace{1cm} (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

\[
\begin{bmatrix}
\text{remaining requirements} \\
\text{effortremaining} \\
\text{teamstrength}
\end{bmatrix} =
\begin{bmatrix}
0 & 1 & 0 \\
0 & $\beta$ & 0 \\
1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
P_t \\
RR_t
\end{bmatrix}
\]

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.

$\begin{align*}
P_{t+1} &= P_t + \epsilon_{pt+1} \\
RR_{t+1} &= -(\gamma +1) P_t + (\gamma +1) RR_t + \epsilon_{rt+1}
\end{align*}$ \hspace{1cm} (15)

$\begin{align*}
P_{t+1} &= 1 \times P_t + 0 \times RR_t + \epsilon_{pt+1} \\
RR_{t+1} &= -(\gamma +1) \times P_t + (\gamma +1) \times RR_t + \epsilon_{rt+1}
\end{align*}$ \hspace{1cm} (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:

remaining requirements $= RR_t$

project velocity $= RR_{t+1} - RR_t$

effortremaining $= \beta \times RR_t + \epsilon_{rt}$

Where $\beta$ represents a factor, which is used for converting remaining requirements into effort remaining; and introduces ontological uncertainty $\epsilon_{\beta}$ for effortremaining related to status vector. The ontological uncertainty represented by random variable $\epsilon_{\beta}$, 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 $2000/4 = 500$ S/ story-points. Then $\beta$ is calculated as follows

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

teamstrength $= P_t$

remaining requirements $= 0 \times P_t + 1 \times RR_t + 0 \times \epsilon_{rt}$ \hspace{1cm} (17)

project velocity $= RR_{t+1} - RR_t$ \hspace{1cm} (18)

effortremaining $= 0 \times P_t + $ $\beta \times RR_t + 1 \times \epsilon_{rt}$ \hspace{1cm} (19)

teamstrength $= 1 \times P_t + 0 \times RR_t + 0 \times \epsilon_{rt}$ \hspace{1cm} (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

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

Software Project Measurement Equation

\[
\begin{bmatrix}
\text{remaining requirements} \\
\text{effortremaining} \\
\text{teamstrength}
\end{bmatrix} =
\begin{bmatrix}
0 & 1 & 0 \\
0 & $\beta$ & 0 \\
1 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
P_t \\
RR_t
\end{bmatrix}
+ \epsilon_{rt}
\]

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 $\varepsilon_p$, $\varepsilon_{\beta}$, and $\varepsilon_u$ related to team-capability $P$, effort remaining, and remaining-requirements $RR$ respectively. The random variables $\varepsilon_p$, $\varepsilon_{\beta}$, and $\varepsilon_u$ follow normal probability distribution. In this paper, we have used the parameter $\Upsilon$ and $\varepsilon_u$ to classify the tracking techniques as follows
- If $\Upsilon$ and $\varepsilon_u$ are zero for all iterations then the model represents a traditional project tracking technique.
- If $\Upsilon$ 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, effort remaining, 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

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

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

Clearly, uncertainty with team capability during first iteration,

$$\text{IndividualDeviation}_{TC1} = \varepsilon_{p1}$$

$$\text{Deviation}_{TC1} = \varepsilon_{p1}$$

Uncertainty with remaining requirements during first iteration,

$$\text{Deviation}_{RR1} = \varepsilon_{u1}$$

$$\text{CumulativeDeviation}_{RR1} = \text{Deviation}_{RR1}$$

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

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

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

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

$$\text{IndividualDeviation}_{TC2} = \varepsilon_{p2}$$

$$\text{Deviation}_{TC2} = \varepsilon_{p1} + \varepsilon_{p2}$$

Uncertainty with remaining requirements during second iteration,

$$\text{Deviation}_{RR2} = -(\Upsilon + 1)\varepsilon_{p1} + (\Upsilon + 1)\varepsilon_{u1} + \varepsilon_{u2}$$

$$= -(\Upsilon + 1)\text{Deviation}_{TC1} + (\Upsilon + 1)\text{Deviation}_{RR1} + \varepsilon_{u2}$$

$$= \text{CumulativeDeviation}_{RR2} = \text{Deviation}_{RR1} + \text{Deviation}_{RR2}$$

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

$$\text{IndividualDeviation}_{TC_{t+1}} = \varepsilon_{p_{t+1}}$$

$$\text{Deviation}_{TC_{t+1}} = \varepsilon_{p_1} + \varepsilon_{p_{t+2}} + .... + \varepsilon_{p_{t+1}}$$

Uncertainty with remaining requirements, individually and cumulatively

$$\text{Deviation}_{RR_{t+1}} = -(\Upsilon + 1)\text{Deviation}_{TC_{t}} + (\Upsilon + 1)\text{Deviation}_{RR_{t}} + \varepsilon_{u_{t+1}}$$

$$\text{Cumulative Deviation}_{RR_{t+1}} = \text{Deviation}_{RR_{t+1}} + ........ + \text{Deviation}_{RR_{t}}$$

### 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 10th iteration after planned duration during 9th iteration.

Table 1 shows the planned and actual project status by showing Remaining_Requirements, Effort_Remaining, Team_Strength, and Project_Velocity at each iteration.
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.
Effort_Remaining as well become negative. That means project actually completed somewhere during the last 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

\[ RR_{t+1} = -(\gamma + 1) \times P_t + (\gamma + 1) \times RR_t + \varepsilon_{ut+1} \]  

(26)

Depending on \( RR_t, P_t, \) and \( \varepsilon_{ut+1}; \) \( RR_{t+1} \) becomes negative.

\[ \text{Effort}_\text{Remaining} = \beta \times \text{Remaining}_\text{Requirements} + \varepsilon_{it} \]  

(27)

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:
Dept. of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece.


[24] D.J.C. Mackay, “Introduction to Monte Carlo Methods,” Department of Physics, Cambridge University, Cavendish Laboratory, Madingley Road, Cambridge, CB30HE , United Kingdom.


