Soft Modeling of Group Dynamics and Behavioral Attributes

Soumya Banerjee[1], **Ajith Abraham**[2], **Sang Yong Han**[2][1] **and Mahanti P.K.**[3]

[1]Dept. of Computer Applications, Institute of Management Studies, India
soumyabanerjee@imsddun.com

[2]School of Computer Science and Engineering, Chung-Ang University, Korea
aith.abraham@ieee.org, hansy@cau.ac.kr

[3]University of New Brunswick, New Brunswick, Canada, pmahanti@unbsj.ca

Abstract: Social networks, religion and culture of human beings play a major role in the day-to-day activities performed by each individual in group oriented missions. The aggregation and inertia in the group are typically important to achieve the goal. A leader being the most dominant and knowledgeable, with leadership qualities, steers movements, thought processes, and actions of the individuals of his/her group. However the psychology of each individual is unique. This complex behavior is often observed in the software development projects, where the cognitive attributes and contribution of programmer's mind are some of the important features to develop a project. This paper proposes a model for the behavior of programmers (software developers) in a development project by incorporating fuzzy logic as a tool. The implication of this model also assists in gaining substantial information about the learning environment of the programmer during the actual implementation and post session of the project and at the same time also helps to evolve the concept of Virtual Project Leader (VPL) for similar projects.

1. Introduction

Group dynamics play a major role during any development process involving psychological and cognitive computing etc. [5]. Human beings, being social animals, like to interact, neglect and communicate (i.e. act in various ways). They do not have the tendency to be idle for long. They must do something or the other, either physically or mentally. The way a group/team acts reflects the way it will perform in the near future. Any software development project could also be considered as a group activity. As a whole, their inertial movements (both physical and behavioral) implicate and affect the software development project. The

[1] Corresponding author email: hansy@cau.ac.kr

movement of group members also put a substantial impact on the resultant [4]. Mathematical models have been solicited to account age difference in hierarchical navigation system, which is monitored mainly on age related differences [13]. Behavioral representation in synthetic forces is the modeling of human behavior as it relates a particular mission. These models should incorporate the spectrum of human biomechanical, physical and psychological parameters, responses and interactions. All such attributes make human behavior highly complex non-linear and adaptable systems. Recently soft computing approaches have been used to address these complex issues [1]. Among the various soft computing paradigms, the hybrid model (combination of neural learning and fuzzy logic) has been already applied in manifold applications successfully. The objective of this paper is to analyze the group dynamics as well as the programmer's mental state behind a software development project, which in turn enables to construct the inference tree indicating the outcome of the project success.

2. Software Development Process and Related Research

The software development process mainly comprises of problem recognition, analysis, feasibility study, design, coding, testing, implementation and maintenance/post implementation. Different software engineering process models have also been proposed to enhance the development process using linear sequential model, prototyping model, RAD model, spiral/win-win spiral model, incremental model, concurrent development model.

All these models claim different methodologies of the development process, though the base remains the same. The models apply different views to facilitate the same Software Development Life Cycle (SDLC) approach. Each of these models requires their individual rations; time constraints, number of developers, cost estimates etc. The process models describes the methodology of implementation but is unable to precise the compatible process to shape the project into success. There are a lot of factors, which monitor any software development project e.g.: deadline constraints, exact output format expected from the development team etc. As the software project is a team effort or a cumulative group activity, a group behavior in the development phase is also functionally one of the most important criterions in the light of behavior modeling of the team members. Substantial works have been solicited relating group behavior of the crowd [2][8]. Crowd is a composition of different people with different mental behavior with/without a goal. Whereas a software development team is an intelligent crowd dedicated towards a particular goal. Therefore broadly this project visualizes a solution space considering typically a software development scenario, where effort is different from project to project basis. In turn this exhibits different mental state of software developers, synchronized with their physical movements in the solution space. Any project of software development has the skill scale marked as follows: Rigorous coding, creativity, analytical ability, patience, endurance level, adaptation to correction (due to Megalomania, Extraordinary sense of superiority complex), etc.

Psychology and behavioral analysis model through simulations has been considered in many significant works [7][8]. The personal Software Process (PSP) is based on the hypothesis that the performance of individual programmers can be improved by applying sound techniques (such as receivers and effort estimation methods) within a defined process (plan, design, code, review, compile, test, post implementation). This entire spectrum depends on the behavior and mental state of the developer. Complete software development life cycle has gained potential to some extent. Ali and Abran [3] used fuzzy logic to measure the software project similarity. To model complex human behavior several attempts have been configured with the help of neuro-fuzzy systems [1]. In software development project, human involvement or involvement of programmers is the basic foundation block. The human i.e. programmer is goal specific, intelligent, may be a good or bad learner, may reciprocate to the situation with different sense of inertia individually. Thus analyzing the behavior of software programmer and its mathematical psychology may lead to a pioneering model, which directly or indirectly explains cognitive soft computing, behavioral study performance measure, etc for a given project. The style of construction of programs is crucially important and it depends on the programmer's ability and thought process. In cognitive psychology [6] thinking process models are often used to explain how we reason about the world around us. Such a model can be seen as a simulation of an object or an abstract concept. Similarly programmers, while creating a system, create a number of mental models of a given system and try to interpret it in the computer language through a program. Speed of the process for the creation of such mental model of given system [12] may vary from programmer to programmer, considering his/her experience and learning factor from the present paradigm [9] or from past projects. Therefore, all the dynamic attributes of the software developer, adept to change, also significantly tell about the success of the project [10]. The effort here is made to incorporate soft computing, typically fuzzy logic based algorithmic and mathematical model to simulate such behavior and thus to reach to the successful or failed status of the project, depending on the mental and behavioral state of the programmer.

3. Modeling Group Dynamics

This section describes a model of group dynamics eventually involved in a standard software development scenario.

1. Problem recognition: Requires excellent analytical ability, healthy group interactions and robust but correct documentation. Group movements would be excessive in and out of the solution space. Relative velocity V_P of individuals with respect to other individuals in the team would change quite frequently. At the same time, mental state feature vector is also of a high magnitude. $V_P = V_P \pm \varphi_P$ ($\varphi_{P\,=}$ considerable variance, V_P = relative velocity of the problem recognition phase). Relative velocity from now on would imply relative velocity of individuals in the solution space.

2. Analysis: After the problem statement is prepared, each developer is expected to understand the problem/work at hand. The task is analyzed, checked, revised for feasibility, cost estimates, time estimates etc. Group movements in this phase could vary from time to time. Relative velocity V_A of employees could remain constant, at a particular interval of time. Mental feature vector also varies, depending upon individual skill set. $V_A = V_A \pm \varphi_A$ (φ_A = negligible deviation, V_A = relative velocity of the analysis phase). While at other intervals it could vary heavily, $V_A = V_A \pm \varphi_A$ ($\varphi_A = \{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_N\}$), where φ_A is a set of considerable deviations.

Phase	**Metrics (probabilistic)**	**Metrics (optimal/near optimal)**
Problem recognition	$V_P = V_P \pm \varphi_P$ @ varied "t"	$V_P = V_P \pm \varphi_P$ @ regular "t"
Analysis	$V_A = V_A \pm \varphi_A$ @ Varied "t", negligible φ_A	$V_A = V_A \pm \varphi_A$ @ varied "t" $\varphi_A \to 0$, $\varphi_A = \{\phi\}$
Feasibility study	Same as analysis	Same as analysis
Design	$V_D = V_D \pm \varphi_D$ @ irregular "t"	$V_D = V_D \pm \varphi_D$ @ occassional "t"
Coding	V_C=K (constant) $\pm \varphi_C$ @ varied "t", negligible φ_C	V_C=K(Constant) $\pm \varphi_C$ @ regular "t", $\varphi_C \to 0$
Testing	V_T=K (constant) $\pm \varphi_T$ @ Varied "t", negligible φ_T $\varphi_T > \varphi_C$	V_T=K (constant) $\pm \varphi_T$ @ Regular "t", negligible φ_T $\varphi_T > \varphi_C$
Implementation	Same as coding	Same as coding
Maintenance	Same as testing	Same as testing

Table 1. Observable velocities of individuals during software development

3. Feasibility Study: Feasibility study won't deviate from the analysis dynamics.
4. Design: The design phase requires a very creative attitude of the developers at hand. So a very low velocity variance is expected with exceptional instantaneous increases. Developers should interact occasionally with each other in the solution space. $V_D = V_D \pm \varphi_D$ (φ_D = considerable variance, V_D = relative velocity of the design phase)
5. Coding: Coding phase requires flat typing in of the codes in a programming language based on the design parameters. Movements around the solution space would be relatively very low, but dynamics of mental vectors of the programmers are very high. V_C = K (constant) $\pm \varphi_C$ (φ_C = negligible variance, V_C = relative velocity of the coding phase)
6. Testing: Testing requires/exhibits considerably few group movements (not negligible). Team members actively interact with associated active agents, to inter-relate their development so that premature errors could be sought out. $\mathbf{V_T = K}$ (constant) $\pm \varphi_T$ (φ_T = negligible variance, $\varphi_T > \varphi_C$, V_T = relative velocity of the testing phase)
7. Implementation: It would correspond to the coding phase.
8. Post implementation/maintenance: It would correspond to the testing phase.
The different phases and the various related metrics are summarized in Table 1.

3.1 Ideal Group Inertia for Software Development

Ideal solution could not be achieved in the practical system. So, one optimal or the most likely solution could be found out. A probable solution space may lead towards an optimal solution. This project proposes the dynamic chart for the group inertia related to software development. Let’s consider each of the velocities, generated by individuals throughout the phases. Such a database would be huge and divided along the repositories of phases and dimensions. Not only mere positional velocities of individual but also accelerative velocities of project metrics or the factors determining the cost estimates of a project are interesting areas of exploration (Here velocity refers to any change in the measurable value of the metrics). The end results would like to indicate such repositories of velocities, behavioral and mental state displacement, calculate the resultant time interval required, to complete a phase and predict whether they are heading towards success, moderate success or failure.

	Velocity notation	**Description**
RELY	V_{RELY}	Rate of change of RELY which might occur at later stages of the development phase.
DATA	V_{DATA}	Rate of change of DATA under unprecedented conditions.
CPLX	V_{CPLX}	Rate of change of CPLX due to unavoidable conditions as erroneous designing of the system .
TIME	V_{TIME}	Rate of change of TIME due to latest customer requirements or emergency.
STOR	V_{STOR}	Rate of change of STOR due to unprecedented breakdowns as power failures, virus attacks etc.
TURN	V_{TURN}	Rate of change of TURN due to dynamic overheads.
ACAP	V_{ACAP}	Rate of change of ACAP due to latest integrative studies.
AEXP	V_{AEXP}	Rate of change of AEXP due to latest accomplishments.
PCAP	V_{PCAP}	Rate of change of PCAP due to recent change in mental state.
LEXP	L_{EXP}	Rate of change of LEXP due to latest accomplishments.
MODP	V_{MODP}	Rate of change of MODP due to momentary popularity of practices etc.
TOOL	V_{TOOL}	Rate of change of TOOL due to unavoidable reasons.
SCED	V_{SCED}	Rate of change of SCED due to emergency or preponed deadlines.

Table 2. Cost drivers and their scalable velocities

3.2 Error Avoidance in Cost Estimates with Reference to COCOMO ‘81

A methodology is proposed for any avoidance of deviational variances from the optimal estimates computed by a particular model. The intermediate version of the COCOMO’81 database [14] is chosen as the basis for our study. The original intermediate COCOMO'81 database contains 63 projects. Each project is described by 17 attributes: the software size is measured in (Kilo Delivered Source Instructions) KDSI; the project mode is defined as organic, semi-detached or embedded and also 15 cost drivers which are generally related to the software environment. Each cost driver is measured using a rating scale of six linguistic values. The assignment of linguistic values to the cost drivers (or project attributes) uses conventional quantization where the values are intervals. In this study, we will consider 13 of the Cost Drivers and their possible scaleable velocities as illustrated in Table 2. The scaleable velocities of the COCOMO'81 attributes are to be obtained from a regressed database of velocities divided along the repositories of the genre of attributes. Such regressed values ought to have an interval of error (ψ) as shown in Figure 1(V_BASE stands for Velocity data base).

- A large positive value of ψ will render the project unsuccessful.
- A moderate positive value of ψ will render the project moderately successful.
- A negligible positive/negative value of ψ will render the project successful.

V_BASE
REGRESSION
$V_R \pm$'ψ'

Figure 1. Error factor from regressed database of velocities

3.3 Analysis Using the Proposed Model

The behavioral presentation in any model of SDLC is primarily monitored through the block of attribute personnel. We experiment this model to represent the mental state of the programmer/analyst capability. Suppose that the developer in the group has successfully completed his assignment, and his contribution influences the project inertia in the following ways, which could be considered as fuzzy linguistic variables:

X_1: Simple/standard behavior during coding

X_2: Exhibited certain reservation towards leader's instructions, not followed SRS properly

X_3: Took extra time, but completed the job, if any lapse, suppressed by group

X_4: Got assistance of other members

X_5: Completed the whole project in a light and daily schedule without extra time assistance from others

X_6: Completed exactly what has been asked for with pleasure

Assume that the developer has the expectation of the group inertia represented by the possibility distribution

$$R_o = (0.9, 0.1, 0.7, 0.3, 0.1, 0.6)$$

We can see from this distribution that the developer expects a positive inertia, may be he/she needs help from the group. If overall, the o/p inertia, for the other members become,

$$E_1 = 0.1/x_1 + 0.8/x_2 + 0.4/x_3 + 0.7/x_5$$

This inertia replicates although relatively strong, unambiguous and clear is rather inconsistent, with the developer's expectation about his group.

Let, $S(M, r) = \max_{x \in Z}[\min(\mu_M(x), r(x))]$

Corresponds to the received message with possibilistic expectations

$S(M_1, r_o) = (0.1,0.1,0.4,0.1)=0.4$.

Because this outcome is contrary to the developer's real expectation, let us assume by virtue of human behavior, he wants to add some distortion.

As equation, $\mu(x) = \mu^S\mu(x)$

Such that it likely, $\mu_1' = 0.4/x_1 + 0.9/x_2 + 0.7/x_3 + 0.4/x$

The already proposed model [11] as such consists of three sub-modules namely activity, productivity and knowledge model. Here, we can map knowledge model into attribute personnel, which clearly demonstrates uncertainty:

$$L_{ij}(\varphi = W_j \qquad K_{ij} \times e - E_{ij}(\varphi - b_{ij}), b_{ij} \langle \varphi$$

$$0, \quad b_{ij} \rangle \varphi$$

where, **$L_{ij}(\varphi)$ =** quantity of gains to knowledge of a developer I by executing a primitive activity of activity j which has a knowledge level 'φ'

b_{ij} = developer i's knowledge level about activity j.

E_{ij} = developer i's downward rate of gain to knowledge by executing activity j.

φ : Required knowledge level to execute the primary activity of activity j.

w_j : Total amount of activity j.

K_{ij} : Maximum quantity of gains to knowledge of developer I by executing activity j.

The K-model presents characteristics of a developer, his willingness to learn, transparency of behavior, sustainability to deadline, abiding to project authority, etc. Considering these aspects an additional complication is introduced when we consider that the software developer may also introduce distortion in the message because with the inconsistency with the expectation, Let

$S(M, r) = \max[\min(\mu_M(x), r(x)]$ (1)

$x \in X$

Correspond the consistency of the receive message which the receiver actually hears as M', where $(\mu_M(x) = (\mu^S{}_M(x))$ (2)

for each, $x \in X$.

The less consistent M with the expectation the less M' resembles M. Since the receiver will be modifying his/her expectations to perform, the new possibilistic expectation structure is given by: $r_1(x) = \min[r_o^{1-S}(x), \mu_M'(x)]$ (3)

for each $x \in X$

Now as measured by (1) the consistency is

$s(M_1,r_o) = \max[0.1, 0.1, 0.4, 0.1) = 0.4$

Since the message is contrary to the developer's expectation; let us assume that he introduces some distortion as we mentioned in, such that the message he hears it:

$M_1' = 0.1/x_1 + 0.9/x_2 + 0.7/x_3 + 0.4/x_5$

Based on this message he modifies his expectation such that

$r_1(x) = \min[r_o^6(x), \mu M_1'(x)]$

for each $x \in X$ or $r_1 = 0.4x_1 + 0.25/x_2 + 0.7/x_3 + 0.25/x_5$

The developer has then experienced greatly diminished negation and his expectation of a simple derisive laughter of the group has given up all hope of the possibility of joy and confidence. Suppose now that in disbelief, the programmer asks the project leader to repeat the judgment and receives the following message,

$M = 0.9/x_2 + 0.4/x_5$

This message is stronger, clearer and less general than the first answer. It's consistency to the developer's new expectation is (of his performance or contribution), $S(M_2, r_1) = 0.25$. Thus the message is highly contrary even to the revised expectation of the developer. So let's suppose that he distorts the message such that he hears, $M_2' = 0.97/x_2 + 0.8/x_5$. His surprise has then diminished the clarity of the message heard and has lead him to exaggerate the degree to which he believes that the project leader has not responded well, he responded with derisive laughter. Now let us suppose that the response which the developer makes the following characteristics from the following set **y,** which is mapped with fuzzy linguistic variables as well:

y_1 = Happily completed, under the guideline of project leader
y_2 = Not happy at the end of the project
y_3 = Surprised about his own good/bad performance
y_4 = Anger/frustration for not to contribute
y_5 = Patient and confident but less happy, about the group support
y_6 = Impatient and asks to change the group
y_7 = Ability to learn more

Let the fuzzy relation $R \in y \times x$ represent the degree to which the programmer plans to respond to a given signal x with a response having the attribute y. Their relationship is given in Table 3

	X_1	**X_2**	**X_3**	**X_4**	**X_5**	**X_6**
Y_1	0.9	0	0.2	0	0	1
Y_2	0	0.9	0.1	0.2	1	0
Y_3	0.1	0.9	0.2	0.9	1	0.3
Y_4	0	0.5	0	0.6	0.7	0
Y_5	0.1	0	0.9	0	0	0.5
Y_6	0	0.3	0.2	0.3	0.4	0
Y_7	0.9	0	0.9	0.3	0	1

Table 3. Relationship between programmer plans to a given signal

$\mu_A(y) = \max [\min (\mu_r (y, x), \mu_M(x)],\ x \in X$

Based on this we can now calculate the response which the developer will make to the message M_2': $A = R_o\ M_2' = 0.9/y_2 + 0.9/y_3 + 0.7/y_4 + 0.4/y_6$. The developer's response therefore will have the characteristics of a great deal of frustration and surprise, a large degree of anger and some impatience.

4. A Decision Tree for Steering a Project to Success

A decision tree as illustrated in Figure 2 is proposed to help in minimizing the 'ψ' value. Special nodes namely ARNs (Administrative Root Nodes) are employed, which govern the weight factors (depicted in Table 4), assigned to their respective child nodes. Weights assigned to branches get incremented by a unit value if a momentary increase in the velocity refresh rate is experienced by a particular ARN node. The responsibility of the ARN now would be to constantly recommend velocities as required so that the 'ψ' value could be kept a minimum. As the prototype of the project is meant for the development of Virtual Project Lead, therefore, the level of decision or D-Tree based on the behavior and mental state of individual members can be configured. Actually, the cumulative effect of such attribute affects the whole project.

Concerned Cocomo'81 attributes	**Maintainable conditions**
Attributes Product	$V_{DATA} < V_{CPLX} < V_{RELY}$
Attributes Material	$V_{STORE} < V_{TURN} < V_{TIME}$
Attributes Personnel	$V_{ACAP} < V_{PCAP} < V_{AEXP} < V_{LEXP}$
Attributes Project	$V_{TOOL} < V_{MODP} < V_{SCED}$

Table 4. Conditions to be maintained for avoiding erroneous deviation

The MARN (Master administrative Root Node), decides priorities of weights in a function COMP (ω, PHASE, n (ω)), ω is a set of weights sent to COMP for comparison. PHASE refers to a particular phase in SDLC.

Here, COMP = (A, $PHASE_X$, 4) $= A_1 < A_2 < A_3 < A_4$

The cardinality factor n(ω) refers to the fact that the proposed model also welcomes any further increase in attribute genres in the COCOMO' 81 models.

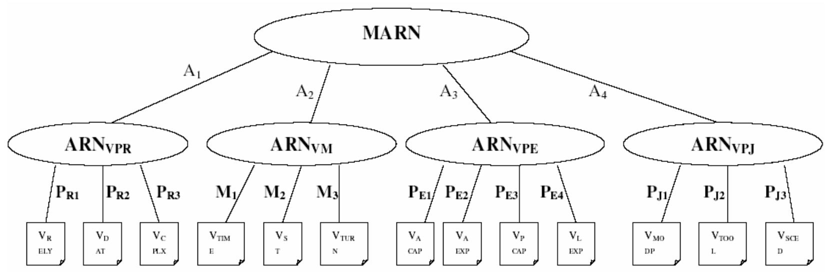

Figure 2. Decision tree approach

5. Conclusions and Future Research

In practice nowadays the IT industry largely follows, object-oriented paradigm, irrespective of the status of the project. In all the cases, programming is the basic implementation of any SDLC, which again depends on individual psychology of the programmer. This paper presented an analytical method to model and analyze such behavior in tune of project leader's dynamics, group dynamics etc. and thus could extrapolate the possibility of success or failure of the project. If the project gets similar attributes in the future, then it may be feasible to apply the acquired behavioral attributes to execute the project. In this context the concept of VPL can be introduced for further development.

Acknowledgements

This research was supported by the MIC (Ministry of Information and Communication), Korea, under the Chung-Ang University HNRC-ITRC (Home Network Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).

References

[1] Gary R., George and Cardullo Frank, Research paper on, Application of Neuro Fuzzy System to Behavioral Representation in Computer Generated Forces, http://citeseer.ist.psu.edu/george99application.html , 1999.

[2] Helbing D., A Mathematical Model for the behavior of Pedestrians, Behavioral Science, 36, 4, pp. 298-310, 1991.

[3] Ali Idri, Alain Abran, A Fuzzy Logic Based Set of Measures for Software Project Similarity: Validation and Possible Improvements, 7th IEEE International Software Metrics Symposium, IEEE Computer Society, pp. 85-97, 2001.

[4] Gray, W. D., Schoelles, M. J. and & Fu W.T., Modeling a Continuous Dynamic Task, Proceedings of the 3rd International Conference on Cognitive Modeling pp. 156-168, 2000.

[5] Kurosu, K., Furuya, T., Nakamura, M. Utsunomiya, H. and Soeda, M., Dynamic and Fuzzy Control of a Group, IEEE International Workshop on Emerging Technologies and Factory Automation, pp. 572-577, 1992.

[6] M. W. Eysenck, M.T. Keane, Cognitive Psychology, A Student's Rawbook, 1990.

[7] Maurizio Morisio, How to Study Individual Programmers, In Proceedings of 22nd International Conference on Software Engineering, Ireland, http://citeseer.ist.psu.edu/684393.html

[8] Musse S.R., Thalman D, Computer Animation and Simulations, Proceedings Eurographics workshop, Budapest Springer Verlag, pp. 39-51, 1997.

[9] Noriko Hanakawa et al., A Software Development Process Simulation Model based on Dynamic changes in Developer's Knowledge Structure", in International Journal of the Annals of S/w Engineering. Vol. 14, pp. 383-406, Oct. 2002.

[10] Noriko Hanakawa, Ken-ichi Matsumoto, Katsuro Inoue, and Koji Torii, A Software Development Simulation Model based on Dynamic Changes in Developer's Knowledge Structure, International Workshop on Software Process Simulation Modeling, England, 2000.

[11] Noriko Hanakawa, Syuji Morisaki and Kenichi Matsumoto, Application of Learning Curve Based Simulation Model for Software Development to Industry, In Proc. 20th International Conference on Software Engineering, pp. 350 – 359, 1998.

[12] Moström, J. E. and D. Carr, Programming Paradigms and Program Comprehension, Proceedings of the 10th Annual Workshop of the Psychology of Programmers Interest Group, UK, pp. 117-127, 1998.

[13] Zaphiris, P., Kurniawan, S.H., Ellis, R.D. Mathematical Formulation of Age Related Differences in Mouse Movement Tasks. In C. Stephanidis (Ed.), Universal Access in HCI, 2003, Lawrence Erlbaum, USA, pp. 917-921. 2003.

[14] Boehm B., Software Engineering Economics. Prentice Hall, 1981.