Balancing Self-directed and Peer-induced Efforts in an Information Technology Collaborative Software Development: A Network Approach

Jaideep Ghosh
Shiv Nadar University, INDIA
jaideep.ghosh@snu.edu.in

Abstract

In a collaboration network environment for software development, it is a highly desirable managerial objective to establish and maintain a stable balance of ongoing engineering activities and workload distributions among the project engineers. Grounded on a theoretical framework and associated simulation models to understand the emergence of this balance, we show that conditioning the development process to remain confined in a stable region of the dynamics could be a delicate affair in real applications. Nevertheless, we demonstrate that it is technically feasible to control ongoing development by suitably restricting relevant parameters of the production dynamics, which amounts to achieving a stable balance between self-directed and peer-induced work efforts expended by the project engineers in building the final software product. Strategies for realizing this condition have important implications for managerial decision-making in collaborative software development.

Keywords

Collaborative software development, production dynamics on collaboration networks, strategies for stable network configuration.

Introduction

Building innovative products in an industrial, collaborative development setting is an increasingly important corporate business model in practically all manners of production operations today. In this model, engineers, from multiple organizations involving a focal firm, a client firm, and any number of consulting firms, possessing a diverse set of skills and expertise, work in the structurally fluid formation of a collaboration network to build the product according to the specified business needs. This style of development is encouraged and promoted by many firms in view of its efficiency and flexibility in the face of changing consumer needs, tastes, and requirements (Ahuja 2000; Booch and Brown 2003). In most practical situations, collaborative development activities are the results of carefully directed efforts in the creation of specialized developmental functions (Stellman and Greene 2006). Networked individuals’ collaborative behavior is structured upon their mutual trust, shared mental models, complementary technical skills, and so on (Amrit and van Hillegersberg 2008; Bhaskaran and Krishnan 2009).

For the management of collaborative engineering operations occurring in multi-institutional and geographically dispersed settings, managers must find the right conditions for balancing self-directed contributory efforts made by the project engineers against their collaborative activities in terms of workload distribution and work compatibility for supporting the most effective product development strategies. The primary purpose of the present study is to study the emergence of this balance. Specifically, we investigate the following research questions: (1) In a collaborative product development situation, how do the self-directed and the peer-induced components of the dynamics govern the behavior...
of the project? (2) Under what conditions is the balance disposed toward a symmetric configuration with approximately similar levels of contributions by the engineers? (3) Is this a very general configuration of the system? If not, can one find a general one for actual applications? The next section explains these concepts and discusses their theoretical and practical relevance.

**Conceptual Foundation of Problem**

To build our research framework, we borrow ideas from two areas of engineering: Control theory and dynamical systems theory. Simply, a dynamical system is a situation where one or more observed variables of interest are changing in time (Barrat et al. 2008). This view helps to conceptualize software development as a dynamical system, whose temporal unfolding is governed, in the present context, by the two components of contributory efforts of the project engineers introduced above. The dynamical variable here is a work amount that an engineer contributes to the project over time, measured in suitable units (discussed later). A vector whose elements are the values of this variable for each engineer in the project characterizes a particular configuration of the collaborative production system. The system dynamics is encoded in this vector. Using the methodology of stability analysis in control theory (Nise 2015), we investigate the possibility of obtaining a configuration of the system that is balanced between the two components. We then formulate strategies using the results of our models.

Three conceptual questions confront us at this point. What exactly is meant by a balance condition in the production dynamics? What are its implications? Why is it a desirable condition for project management? Unlike run-of-the-mill or minimally designed software, innovative products do not commonly have fixed requirements (Espinosa et al. 2007; Herbsleb and Moitra 2001). As novel features are continuously researched and developed, the software engineers engage in autonomous as well as in collaborative involvement in building the final product: They make self-directed creative efforts, which are largely uninfluenced by peer influence or intervention. In addition, they also incorporate intermixed ideas obtained through their collaboration network contacts (Ahuja 2000). However, for project management, a tension arises in the attempt to balance these two contrasting characters of work efforts. Ascertaining what degree of individual efforts can be effectively reconciled with what degree of collaborative engagements under prevailing development conditions is a difficult proposition. From the perspective of control theory, the critical significance of the balanced configuration of the system lies in the fact that it brings in a regulated uniformity between the self-directed and peer-induced efforts of the project developers. More importantly, this configuration must be stable over time, so that smoothness of production is sustained. In real applications, this behavior is emergent and cannot be asserted a priori. Additionally, in the combined dynamics of the components, the balanced configuration may come out to be symmetric, as when each engineer expends approximately the same level of efforts in their work. More generally, however, the balanced configuration is non-symmetric, which happens when the engineers expend unequal amounts of work efforts.

As for the last question, we note that the peer-induced, associative work is governed by the search, transfer, and intermixing of ideas, skills, domain knowledge, and expertise of the networked engineers (Brown and Duguid 1991; Dodgson 1993). Supplementing these activities is each individual engineer’s own creative efforts (Durlauf 2001; Sparrowe et al. 2001). These are self-directed, in that they are expressed solely by autonomous efforts and do not require motivation or guidance from peers to be functional. In this duality, the balance condition ensures that the two types of efforts are properly harmonized with each other. Without this essential control, such efforts, no matter how innovative, may quickly become quickly unrealistic, impractical, or just chaotic (Amrit and van Hillegersberg 2008; Brown and Duguid, 1991).

**Empirical Setting**

This work originated in the present author’s advisory involvement in the development environment of a multinational software product firm (designated as Cx). In its different projects, the software engineers are dispersed in development centers across a number of countries. Project managers maintain an exceedingly detailed record-keeping system for tracking continuous work progress in the form of incremental work records in a centralized database, called the codebase. In a typical project, the software product is built in a number of modules, to the units of which team members incrementally contribute. An
engineer works both autonomously as well as in collaboration with co-workers. The collaborative ties between the engineers are established early in the project.

We conceptualize the development system as a collaboration network of project engineers, constructed out of cross-sectional data for a time horizon of $T = 289$ days, with 53 engineers for a total number of 227 singly counted, non-directional, weighted ties between pairs of engineers. A tie linking a pair has a strength proportional to the number of common work units in the modules in which they have collaborated. The project comprises of seven teams (team sizes): Specifications (4); Requirements (7); Design (9); Programming (12); Testing (10); Integration (also called Implementation) (7); Architecture (4). In the codebase, there are instances of collaborative connections between pairs from the following teams: Specifications & Requirements; Requirements & Design; Design & Programming; Programming & Testing; Programming & Integration; Design & Integration; Design & Architecture; Programming & Architecture; Requirements & Testing; Requirements & Architecture; and Specifications & Testing.

<table>
<thead>
<tr>
<th>Date</th>
<th>Phase</th>
<th>Unit</th>
<th>Module</th>
<th>Sub-module</th>
<th>Contribution Type</th>
<th>Contributor Id</th>
<th>Contributed Amount (%)</th>
<th>Module Completed (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>15-05-13</td>
<td>Design-code</td>
<td>UI-26</td>
<td>Order</td>
<td>Item selection</td>
<td>Individual</td>
<td>AX317</td>
<td>0.3</td>
<td>0.8</td>
</tr>
<tr>
<td>27-05-13</td>
<td>Design-code</td>
<td>SRV-29</td>
<td>Order</td>
<td>Server call handler</td>
<td>Multiple</td>
<td>FT441 GC386</td>
<td>0.9</td>
<td>0.6</td>
</tr>
<tr>
<td>28-05-13</td>
<td>Design-code</td>
<td>SRV-29</td>
<td>Order</td>
<td>Server call handler</td>
<td>Multiple</td>
<td>FT441 GC386</td>
<td>0.8</td>
<td>0.6</td>
</tr>
<tr>
<td>29-05-13</td>
<td>Design-code</td>
<td>SRV-29</td>
<td>Order</td>
<td>Server call handler</td>
<td>Multiple</td>
<td>FT441 RA227</td>
<td>0.6</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Table 1. A Simplified Snapshot of Codebase

Table I shows a simplified version the codebase of a typical project. The Contributed Amount column records the amounts of the self-directed and peer-induced efforts contributed by the software engineers to work units. These are computed as the cumulative total increasing from a predetermined initial value. The Contribution Type column indicates whether the contribution is self-directed (“Individual”) or peer-induced (“Multiple”). Of course, both of the efforts are made concurrently in real time. The Module Completed column accounts for all self-directed or peer-induced contributions to a module over a definite period (typically, six workdays a week). Most information in the codebase is updated at least once a day every workweek (note the Date field in Table 1). In this work, we utilize 289 days of data from the codebase, which marks the study’s time horizon ($T = 289$). Further, to have sufficient variations in the data, we use an incremental time window of one day ($\Delta t = 1$) as the value of a single time step in our network simulations. Any single structure of the network is constructed at a fixed cross-section of time.

**Study Assumptions**

1. Clients are part of C’s external environment. They provide product specifications at some initial time but do not interfere, thereafter, with the project’s internal processes. Consequently, at least for each release, requirement specifications are frozen.
2. When considering the work differentials of an engineer, the work contributions from all others are linearly additive. This approximation scheme works well for relatively long horizons.
3. Ties once formed in the network do not disappear. This is a reflection of the large-scale network topology, relevant to one structure that is fixed over one cross-section of time.
4. An individual’s self-directed efforts are distinct from their peer-induced efforts. Thus, any feedback between the two components of the dynamics is ignored. (See Conclusions section).
5. The behavior of each software engineer in the project is cooperative in time. Thus, collaborative behavior, at least in the present context, is not strategically exploitative. This assumption, grounded on the theory of inter-personal trust, is known as the reciprocity assumption and is well supported by earlier work in technological collaboration (Ahuja 2000; Cox et al. 1997).

**Theoretical Framework**

**Network Characterization**
As described before, our theoretical framework conceptualizes collaborative software development as a system, whose dynamics is executing on the collaboration network of \( n \) engineers with \( m \) collaborative ties linking pairs of engineers. This network is represented by an \( n \times n \) weight matrix \( W \) with elements \( W_{ij} = 1, 2, ..., \) if engineers \( i \) and \( j \) are associated by means of co-development work in a number 1, 2, ... of areas (or sub-areas) of the modules designed in the project; \( W_{ij} = 0 \) otherwise. The directionality of the tie linking engineers \( i \) and \( j \) does not matter. Also, \( W \) is a symmetric matrix: \( W_{ij} = W_{ji} \).

**Software Development Dynamics on Network**

The dynamical variable \( S_i(t) \) represents the fractional amount of work contributed by engineer \( i \) to the project. Its rate of change results from two independent dynamical components: (1) **self-directed dynamics** \( \psi(S_i) \) characterizes engineer \( i \)’s autonomous contribution to a work unit in a module; (2) **peer-induced dynamics** \( \chi(W,S) \) characterizes work that engineer \( i \) contributes to a work unit in association with co-developers (Barrat et al. 2008; Newman 2010). The complete dynamics has the form: \( \frac{dS_i(t)}{dt} = \psi(S_i) + \sum_j W_{ij} \varphi(S_i, S_j) \). The second term on the right embodies linearly additive collaborative contributions, where \( \varphi(S_i, S_j) \) is a 2-point function that embodies the pairwise collaborative interaction of \( i \) and \( j \). For generality, we assume that \( \varphi(S_i, S_j) \) is a non-symmetric interaction, meaning \( \varphi(S_i, S_j) \neq \varphi(S_j, S_i) \). A balanced configuration is described by the vector \( \Theta = \{ S'_i \} \) for which \( \frac{dS_i(t)}{dt} = 0 \), yielding the condition: \( \psi(S'_i) + \sum_j W_{ij} \varphi(S'_i, S'_j) = 0 \). Taylor-expanding both \( \psi \) and \( \varphi \) around \( \Theta \), with \( \eta \) serving a small perturbation around \( \Theta \) as \( S_i = S'_i + \eta_i \), we obtain \( \frac{d\eta_i}{dt} = \eta_i \psi(S'_i) + \eta_i \sum_j W_{ij} \varphi_a(S'_i, S'_j) + \sum_j \eta_j W_{ij} \varphi_B(S'_i, S'_j) = \rho_i \eta_i + \eta_i \sum_j \sigma_{ij} W_{ij} + \sum_j \tau_{ij} W_{ij} \eta_j \), where the parameters are defined as \( \rho_i = \psi(S'_i); \sigma_{ij} = \varphi_a(S'_i, S'_j); \tau_{ij} = \varphi_B(S'_i, S'_j) \).

**Symmetric Balance**

The balanced configuration is symmetric across all engineers in the network if each contributes approximately equal amounts of work efforts (in appropriate units of measurement). This condition yields a balanced configuration realized by the vector \( \Theta = \{ S'_i \} = S'_i, \forall i = 1, ..., n \), where \( \psi(S'_i) + \varphi(S'_i, S'_i) \sum_i W_{ij} = 0 \). This holds when \( \varphi(S'_i, S'_i) = 0 = \psi(S'_i) \). The model parameters are given by \( \rho_i = \rho = \psi(S'_i); \sigma_{ij} = \sigma = \varphi_a(S'_i, S'_i); \tau_{ij} = \tau = \varphi_b(S'_i, S'_i) \). The network dynamics becomes: \( \frac{d\eta_i}{dt} = (\rho + \sigma \zeta) \eta_i + \tau \sum_j W_{ij} \eta_j \), where \( s_i = \sum_{k=1}^n W_{ik} \) is the strength of engineer \( i \), and \( \zeta = \frac{1}{n} \sum_{k=1}^n s_k \) is the average network strength. The strength metric of engineer \( i \) is proportional to the number of work units in which the engineer has collaborated.

**Differential Coupling**

It is reasonable to assume that \( \varphi(S_i, S_j) \) differentially couples the contributions of engineers \( i \) and \( j \). Thus, as engineer \( j \) keeps investing larger and larger amounts of efforts in a module unit, the collaborating engineer \( i \) will, in turn, expend larger amounts of efforts in the same unit. The coupling will then equal the deficit amount of their individual efforts as \( \varphi(S_i, S_j) = \omega(S_i) - \omega(S_j) \), where \( \omega(S) \) is a 1-point function embodying the collaborative dynamics effect on a single engineer. This gives \( \frac{d\eta_i}{dt} = \rho \eta_i + \sigma \sum_j (s_i \delta_{ij} - W_{ij}) \eta_j \), where \( \delta_{ij} \) is the Kronecker delta, and \( \rho = \psi'(S'_i); \sigma = \varphi_a(S'_i, S'_i); \tau = \varphi_b(S'_i, S'_i) \). Writing \( \mathbb{G}_{ij} = s_i \delta_{ij} - W_{ij} \) as the elements of a generalized graph Laplacian matrix \( \mathbb{G} \) (Anderson and Morley 1985; Olfati-Saber 2006), we now obtain \( \frac{d\eta_i}{dt} = \sum_j (\rho \delta_{ij} + \sigma \mathbb{G}_{ij}) \eta_j \). In matrix form, we have \( \frac{d\eta}{dt} = \mathbb{M} \eta \), where the matrix \( \mathbb{M} = \rho \mathbb{I} + \sigma \mathbb{G} \), and \( \mathbb{I} \) is the unit matrix. \( \mathbb{M} \) has elements \( M_{ij} = \rho \delta_{ij} + \sigma \mathbb{G}_{ij} \). The eigenvalue \( \omega_q \) of \( \mathbb{M} \) is given by \( \omega_q = \rho + \sigma \lambda_q \), where \( \lambda_q \) is the eigenvalue of the generalized Laplacian \( \mathbb{G} \).

**Network Stability Configurations**

The desired stability of a symmetric balanced configuration of the network is given by \( \omega_q = \rho + \sigma \lambda_q < 0 \). As \( \lambda_q \)'s are known to be positive semi-definite (Olfati-Saber 2006), we have \( \rho = \psi'(S'_i) < 0 \). The master stability condition, then, is \( \frac{1}{\lambda_q} > -\epsilon \), where \( \epsilon = \frac{\sigma}{\rho} = \frac{\omega'(S'_i)}{\psi'(S'_i)} \). Here, we are guaranteed of the stability of the configuration when this condition holds for the largest eigenvalue \( \lambda_{tar} \) of \( \mathbb{G} \). The network’s actual physical
structure is encoded in $\mathbf{G}$ through the weight matrix $\mathbf{W}$, which is reflected in the quantity $\lambda_{\text{tar}}$. By contrast, the dynamics of the network is embedded only in the two functions $\omega$ and $\psi$.

**General Case of Non-Symmetric Balance**

The symmetric situation is both extremely delicate as well as highly specialized. Most commonly, one encounters in most real applications a non-symmetric configuration $\bar{\Theta} = \{S_1^*, S_2^*, \ldots, S_n^*\}$. The balance condition takes the form $\psi(S_i^*) + s_i \omega(S_i^*) - \sum_j W_{ij} \omega(S_j^*) = 0$. Proceeding exactly as in the symmetric case, we obtain $\frac{dn_i}{dt} = \rho_i \eta_i + \sum_j \sigma_{ij} W_{ij} + \sum_j \tau_{ij} W_{ij} \eta_j$, where the parameters are given by $\rho_i = \psi'(S_i^*)$, $\sigma_{ij} = \varphi_a(S_i^*, S_j^*) = \omega'(S_i^*) = A_i$, $\tau_{ij} = \varphi_b(S_i^*, S_j^*) = -\omega'(S_j^*) = -B_j$. Calculations lead to the final result: $\frac{dn_i}{dt} = \rho_i \eta_i + \sum_j \tilde{G}_{ij} \eta_j$, where $\tilde{G}_{ij} = s_i A_i \delta_{ij} - B_j W_{ij}$ is a modified form of $\mathbf{G}$. In matrix form, we write $\frac{d\mathbf{n}}{dt} = \mathbf{N} \mathbf{\eta}$, where the matrix $\mathbf{N}$ has elements $\mathbf{N}_{ij} = \rho_i \delta_{ij} + \tilde{G}_{ij}$. The stability condition connects the leading eigenvalue $\lambda_{\text{tar}}$ of $\mathbf{G}$ with the dynamical functions of the self-directed and the peer-induced efforts for the balanced configuration $\bar{\Theta}$ of the network.

**Computational Details of Models and Simulations**

**Self-directed model of dynamics**

The easiest way to discover a model of the self-directed component of the dynamics is to plot an empirical time series of the growth of the self-directed efforts of the engineers. In our case, using data from the project’s codebase, we plot the time series in Figure 1. Based on the plot, we simulate the self-directed component by a phase-plane function of the form $\psi(S_i) = \alpha(1 - S_i)$, where $\alpha > 0$ is a constant. In this case, the network tends to settle down to $S^* = 1$, which makes $\psi(S^*) = 0$. Figure 2 depicts the qualitative aspects of the dynamics. In the region $S \in (0,1)$, $\psi(S) > 0$, and the flow is directed to the right toward $S^* = 1$. In the region $S > 1$, $\psi(S) < 0$, and the flow is directed to the left toward $S^* = 1$. Thus, all flows carry the system toward $S^* = 1$, which is both locally and globally stable.

![Figure 1. Time-series of self-directed efforts (unscaled)](image1.png)

![Figure 2. Self-directed component in phase plane](image2.png)

**Peer-induced models of dynamics**

In this scenario, the 2-point function can be realized in the representation of the following two models.

**Model 1.** With collaborative interaction that is not strategically opportunistic, an individual’s peer-induced contribution can be modeled by a function $\omega(S_i) = \beta \frac{S_i}{1 + S_i}$, where $\beta > 0$ is a constant. Figure 3 displays the form of $\omega(S)$. Although $S^* = 0$ is the balance in this case, it is inherently unstable. The 2-point function has the form: $\varphi(S_i, S_j) = \beta \frac{S_i S_j}{(1 + S_i)(1 + S_j)}$, which is displayed in Figure 5. The nontrivial balance corresponds to $S^* = 1$, and calculations show that $\epsilon = -\frac{\beta}{4\alpha} < 0$, since $\alpha, \beta > 0$. Our first simulation is based on the condition: $\lambda_{\text{tar}} < \frac{4\alpha}{\beta}$. 

![Figure 3. Peer-induced efforts](image3.png)
Model 2. In this model, the individual contribution of $i$ working in collaboration with $j$ is described by the function $\omega(S_i) = S_i(y + S_j - S_i)$, where $y > 0$ is a constant. This function provides a synergistic relationship between two engineers: Given $S_j$, engineer $i$’s contribution is a quadratic function of the effort level $S_i$ that is zero when $S_i = 0$ and when $S_i = y + S_j$, and reaches a maximum value of $\frac{1}{4}(y + S_j)^2$ in between at the point $\hat{S}_i = \frac{1}{2}(y + S_j)$. Figure 4 displays the form of $\omega(S_i)$ for $S_j$.

The 2-point function has the form $\varphi(S_i, S_j) = (S_i - S_j)(y - S_i - S_j)$, shown in Figure 6. The balance for the self-directed dynamics is given by $\hat{S}_i = \frac{1}{2}(y + S_j)$. Calculation show that $\omega'(S^* = 1) = y - 1$, $\psi'(S^* = 1) = -\alpha$, and $\epsilon = -\frac{y - 1}{\alpha}$. The master stability condition of the balance is $\lambda_{\text{lar}} < \frac{\alpha}{y - 1}$, where $\alpha, y > 0$. In addition, the Laplacian, being a symmetric matrix, has real eigenvalues, which are also non-negative (Anderson and Morley 1985). This imposes an additional restriction $y > 1$. Thus, the simulation of this model is based on the condition given by $0 < y < \left(\frac{1}{\lambda_{\text{lar}}}\right)\alpha$.

Monte Carlo simulations$^2$

We look for stability regions in the parameter spaces $(\alpha, \beta)$ and $(\alpha, y)$. Our strategy here is to perform Monte Carlo simulations using a non-parametric bootstrap estimation procedure based on actual sample data (Efron and Tibshirani 1993). We estimate parameters using a merit function $\chi^2 = \sum_{i=1}^{\hat{S}} (\frac{S(t_i) - S_{\text{act}}(t_i)}{\sigma_{S_i}})^2$ by minimizing the squared-errors between simulated $S_i$ and their observed values $S_{\text{act}}$ at discrete time points $(t_1, ..., t_k)$. The parameters are estimated using the Levenberg-Marquardt method (Press et al. 1992). Parameters accuracy bounds are obtained in Monte Carlo steps by the percentile method (Joshi et al. 2006): If $\hat{\theta}^{(\alpha)}$ stands for the $100(1 - \alpha)$ percentile of the $B$ bootstrap replications, then the confidence interval is given by $(\hat{\theta}_i, \hat{\theta}_i) = (\bar{\theta}, \bar{\theta}^{1-\alpha})$.

Results

$^1$ If an opportunistic game were played by engineers $i$ and $j$, this state would be the best response function of engineer $i$ to $j$: $\beta_i(S_j) = \frac{1}{2}(y + S_j)$.

$^2$ Details of this procedure are not included in this paper.
**Simulations of Models 1 and 2**

Figure 9 shows the results of our simulations of model 1. In the designated parameter space, the line demarcates the regions of stability and instability of the network's balance configuration. The region lying below the borderline is characterized by lines having slopes smaller than that of the borderline, which is approximately 0.1. In model 2, the borderline shown in Figure 10 has a smaller slope than that of the one in model 1, signifying the even smaller size of the stability region. This indicates that the choice of the relative significances of the $\alpha$ and $\gamma$ parameters is rather delicate in practice.

![Figure 9. Simulation of model 1](image1)

![Figure 10. Simulation of model 2](image2)

**Monte Carlo Simulations**

The bootstrapped Monte Carlo results are displayed in Figures 11 and 12 for models 1 and 2 respectively. In Figure 11, $\alpha$ and $\beta$ are highly correlated, with a correlation coefficient of $r \approx 0.91$, significant at 95%. The figure also shows three different confidence regions, at the same value of the confidence level. The two vertical lines enclose a band representing the 95% confidence interval for $\alpha$ independent of $\beta$, and the two horizontal lines enclose a 95% confidence interval for $\beta$, independent of $\alpha$. The ellipse shows a 95% confidence interval for the joint distribution of $\alpha$ and $\beta$ (Press et al. 1992). Figure 12 also shows these plots for model 2, where $\alpha$ and $\gamma$ are seen to be highly correlated, with a correlation coefficient of $r \approx 0.92$, significant at 95%. Based on the average value $\bar{\theta}$ of a model parameter, the range $\Lambda$ and shape $\Gamma$ of the confidence interval are given by $\Lambda = \bar{\theta}_u - \bar{\theta}_l$ and $\Gamma = \frac{\bar{\theta}_u - \bar{\theta}}{\bar{\theta}_l - \bar{\theta}_l}$ (Joshi et al. 2006). A $\Gamma$ exceeding 1.0 signifies a smaller distance from $\bar{\theta}$ to $\bar{\theta}_l$ than from $\bar{\theta}_u$ to $\bar{\theta}_l$. For a normal distribution symmetric about $\bar{\theta}$, $\Gamma$ equals 1.0. In both models, $\Gamma \approx 0.93$ for about 95% of the cases, indicating that the dynamics does not deviate significantly from linearity in the model parameters (details not included).

**Discussions**

The present framework and some its associated models have actually been used in three projects of $C_x$. A detailed description of this procedure lies outside the purview of this brief communication, which focuses primarily on the conceptual aspects of the problem. However, it would not be entirely out of place to
briefly outline the method of practical implementation of our framework. Importantly, this outline will serve to illustrate the primary contribution of the present study, which is how to make decisions in practical applications of collaborative product development regarding the discovery of the desired balanced configuration of the system.

The first step in a real-time application of the framework is to fix an initial time horizon for assessment of the project dynamics. The second step is to collect sufficient network data for this time window. The third step is to calibrate the models (that is, fixing the relevant parameter confidence intervals) using Monte Carlo simulations exactly as described above, and then to assess whether or not the current network configuration lies in the region of stable balance (using the stability conditions given above). Armed with this knowledge, the final step is to compute all residuals from the theoretical stable balance configuration and to formulate corrective strategies (discussed below). This entire process can be repeated and assessments made as many times as desired by executing the same simulations in different time windows.

In my experience with the use of the present framework in real collaborative software development projects, it is quite unlikely that one encounters a truly symmetric balanced configuration of the system in most common collaborative software development projects. Work efforts of the engineers differ widely in their expertise, experiences, levels of maturity, work styles, current task complexities and loads, educational qualifications, and so forth (Espinosa et al. 2007; Sparrowe et al. 2001). As an exception, one may be lucky enough to find a symmetric balance in some highly specialized product development situation (and that too for only a brief period of time). In most common industrial applications, one should look for non-symmetric balance.

In order to formulate strategies to drive product development toward a non-symmetric balanced configuration, trial simulations must first be run over short time windows of recent past to discover the prevailing conditions in the system. These results are then used to guide the strategy formulation process for subsequent time periods. We now describe two important scenarios that can be used as illustrations to motivate the general procedure for formulating strategies in real applications.

First scenario. Suppose that simulations executed on a trial window reveal that a stable balance is unrealizable under current conditions, because numerous self-directed, scattered efforts are being made by the project engineers without concurrently supporting group activities. This is a clear indication that the balance is disposed toward the $\alpha$-weights. To rectify the problem, we provide an example of a strategy that has been used successfully in a real project. The strategy is to form a number of small informal groups (not official project teams) of collaborating engineers. In each group, one or two selected individuals are experienced members possessing a wide range of technical skills and expertise, while the other members in the group are relatively less experienced but technically adept. The more experienced members are expected to generate the bulk of the ideas in the assigned tasks, which the less experienced ones are instructed to physically implement. The reason why this strategy works is that, the guidance and the directions provided by the experienced members help to consolidate and coordinate ideas among the less experienced members in the groups, reducing the $\alpha$-weights sufficiently and raising the possibility of realizing stable balance in the near future.

Second scenario. Suppose the initial simulations reveal that the parameter vector lies in a particularly thin band around the boundary line between the stable and unstable areas in parameter space. This is a tricky situation to manage in practice, because, under this condition, stable balance cannot be uniformly maintained. If the parameter vector lies in the stable region, then the current configuration is temporarily stable, although not uniformly so: A little change in the existing conditions might destabilize the system. On the other hand, if the parameter vector lies in the unstable region, then the system is uniformly unstable. To tackle this situation, strategies must be oriented toward bringing the parameter vector away from the boundary and deep into the stable region. A successful example of a strategy used in a real project consists of forming small groups of two to three engineers of comparable but complementary skills to work collaboratively on work units that require cross-functional expertise. Assuming the $\alpha$-weights of the engineers are of comparable magnitude, attempts should be made to reduce the corresponding $\beta$- or $\gamma$-weights (as the case may be) well below the value of the theoretical boundary vector. Physically, this strategy gives emphasis to more concentrated, self-directed efforts in the concerned units and recommends collaborative action when work unit integration is called for. The helps the network to reach a stable balance configuration, without making drastic changes in individual engineers’ work-differentials.

Conclusions and Future Directions
Although the present framework has been applied successfully in real collaborative software development projects, there are a few deficiencies in the formulation. The first limitation lies in the choice of the time horizon of the dynamics. The theoretical results are best for long time frames, in which it is much easier to find stable configurations that can be sustained over time. A related issue is the lack of decay of the tie distribution in the network. Unless a human relationship is routinely nurtured, its strength tends to decay over time, and it does not make much sense to keep dormant ties in the network. The tie strength should, more accurately, be formulated as a function of time. Next, the software development projects of Cₓ exemplified in this work are not grossly influenced by the administrative policies of other consulting firms or owned subsidiaries of Cₓ involved in co-development of the software product. Although external consultants are brought in to work on the project, the decisions pertaining to both capital and labor investments are still made by the managers of Cₓ. Consequently, exogenous policy-related complexities are much reduced in these projects. In many real-world collaborative software development projects, the administrative situation and external control may not be so simple. In such cases, the network must be constructed differently. Finally, our framework is constrained by the absence of actual feedbacks operating between the self-directed and the peer-induced components. In actual practice, self-directed efforts are not entirely independent of the concurrently unfolding peer-induced efforts and vice versa. Each individual developer is likely to be influenced by the activities of others; the teams, in turn, are guided by the self-directed contributions made by individual engineers. Therefore, in a more general setting, there should be interactive terms in the dynamics. In the context of the projects of Cₓ, I was unable to establish the existence of this feedback simply on the basis of the secondary data I had collected from the project codebases. Ongoing work at this time entails incorporating some of these enhancements in the present framework.

REFERENCES


