A.1 Formulation of the Time Minimization Problem

Minimize $\Lambda$

Subject to:

\[
\sum_{i \in M_l} \left[ D_{ij} X_{ij} + \sum_{i \in f} \beta_{ijl} \left( 0.5D_{ijl} - D_{ij} \right) \right] + \sum_{l=1}^{S} 0.5 f_{jl} q_{jl} + \sum_{(i,z) \in N_i} \left[ \frac{\lambda^z_{i}}{2} \left( w_{i,z} x_{ij} + w_{i,z} x_{ij} \right) + \frac{\lambda^p_{z}}{4} \left( w_{i,z} x_{ij} + w_{i,z} x_{ij} \right) + \frac{\lambda^\alpha_{z}}{3} \left( w_{i,z} x_{ij} + w_{i,z} x_{ij} \right) - \right. \\
+ \left. \sum_{l=1}^{S} \alpha_{l} \left( \frac{\lambda^z_{i}}{2} \left( w_{i,z} x_{ij} C_{ij} + w_{i,z} x_{ij} C_{ij} \right) + \frac{\lambda^p_{z}}{4} \left( w_{i,z} x_{ij} C_{ij} + w_{i,z} x_{ij} C_{ij} \right) + \frac{\lambda^\alpha_{z}}{3} \left( w_{i,z} x_{ij} C_{ij} + w_{i,z} x_{ij} C_{ij} \right) \right) \right] \leq T_i
\]

$\forall j = 1,2,\ldots,S$

\[
T_i + \sum_{i \in M_j} \left[ D_{ij} X_{ij} + \sum_{i \in f} \beta_{ijl} \left( 0.5D_{ijl} - D_{ij} \right) \right] \\
+ \sum_{(i,z) \in N_i} \left[ \frac{\lambda^z_{i}}{2} \left( w_{i,z} x_{ij} + w_{i,z} x_{ij} \right) + \frac{\lambda^p_{z}}{4} \left( w_{i,z} x_{ij} + w_{i,z} x_{ij} \right) + \frac{\lambda^\alpha_{z}}{3} \left( w_{i,z} x_{ij} + w_{i,z} x_{ij} \right) - \\
+ \sum_{l=1}^{S} \alpha_{l} \left( \frac{\lambda^z_{i}}{2} \left( w_{i,z} x_{ij} C_{ij} + w_{i,z} x_{ij} C_{ij} \right) + \frac{\lambda^p_{z}}{4} \left( w_{i,z} x_{ij} C_{ij} + w_{i,z} x_{ij} C_{ij} \right) + \frac{\lambda^\alpha_{z}}{3} \left( w_{i,z} x_{ij} C_{ij} + w_{i,z} x_{ij} C_{ij} \right) \right) \right] \leq T_j
\]

$\forall j = 1,2,\ldots,S$

Similarly, there are sequential constraints for groups 3, 4, \ldots, $G - 1$. 


Finally,

\[ T_{G-j} + \sum_{i \in M_G} \left[ D_{ij}^p X_{ij} + \sum_{i \neq j} \beta_{ijl} \left( 0.5D_{ijl}^p - D_{ijl}^q \right) \right] \]

\[ + \sum_{(i-j) \in N_G} \left[ \frac{\lambda^s_{x_{ij}}}{2} \left( w_{x_{ij}} x_{ij} + w_{x_{ij}} y_{ij} \right) + \frac{\lambda^p_{x_{ij}}}{4} \left( w_{x_{ij}} x_{ij} + w_{x_{ij}} y_{ij} \right) + \frac{\lambda^o_{x_{ij}}}{3} \left( w_{x_{ij}} x_{ij} + w_{x_{ij}} y_{ij} \right) - \lambda^s_{x_{ij}} \right] \leq \Lambda \]

\[ \forall j = 1, 2, \ldots, S \]

Effort Budget Constraint

\[ \sum_{i=1}^{m} \sum_{j=1}^{S} D_{ij}^p X_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{S} \sum_{l=1}^{S} \sum_{j=1}^{S} \beta_{ijl} \left( D_{ijl}^p - D_{ijl}^q - D_{ijl}^r \right) + \sum_{j=1}^{S} \sum_{i=1}^{m} f_{ij} q_{ij} \]

\[ + \sum_{i=1}^{m-1} \sum_{j=1}^{S-1} \left[ \lambda^s_{x_{ij}} w_{x_{ij}}^s + \lambda^p_{x_{ij}} w_{x_{ij}}^p + \lambda^o_{x_{ij}} w_{x_{ij}}^o - \sum_{j=1}^{S} \alpha_x \left( \lambda^s_{x_{ij}} w_{x_{ij}}^s + \lambda^p_{x_{ij}} w_{x_{ij}}^p + \lambda^o_{x_{ij}} w_{x_{ij}}^o \right) \right] p_{ij} \leq B \]

In addition, the module assignment constraints are same as those in the effort minimization model.

A.2 ADDITIONAL LINEARIZATION VARIABLES FOR BOTH EFFORT MINIMIZATION AND TIME MINIMIZATION PROBLEMS

\[ y_{ij}^{s} = 1 \quad \text{if module } i \text{ is developed using solo approach; 0 otherwise} \]

\[ y_{ij}^{p} = 1 \quad \text{if module } i \text{ is developed using pair approach; 0 otherwise} \]

\[ e_{ij}^{k} = y_{ij}^{s} y_{ij}^{p} ; k = s, p ; l = s, p \]

\[ U_{iklj}^{l} = w_{iklj}^{l} x_{ij} ; l = s, p \]

\[ V_{iklj}^{l} = U_{iklj}^{l} C_{iklj} ; l = s, p \]

\[ H_{iklj}^{l} = w_{iklj}^{l} C_{iklj} ; l = s, p \]

A.3 ADDITIONAL LINEARIZATION CONSTRAINTS FOR BOTH EFFORT MINIMIZATION AND TIME MINIMIZATION PROBLEMS

\[ \beta_{ijl} \leq q_{ijl} \quad \forall i = 1, 2, \ldots, m ; j = 1, 2, \ldots, S - 1 ; l = j + 1, j + 2, \ldots, S \]
\[ \beta_{ij} \leq X_{ij} \quad \forall i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, S - 1; \quad l = j + 1, j + 2, \ldots, S \]

\[ \beta_{ij} \leq X_{il} \quad \forall i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, S - 1; \quad l = j + 1, j + 2, \ldots, S \]

\[ \beta_{ij} \geq X_{ij} + X_{il} - 1 \quad \forall i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, S - 1; \quad l = j + 1, j + 2, \ldots, S \]

\[ C_{ij} \leq X_{ij} \quad \forall i = 1, 2, \ldots, m - 1; \quad z = i + 1, \ldots, m; \quad j = 1, 2, \ldots, S \]

\[ C_{ij} \leq X_{ij} \quad \forall i = 1, 2, \ldots, m - 1; \quad z = i + 1, \ldots, m; \quad j = 1, 2, \ldots, S \]

\[ C_{ij} \geq X_{ij} + X_{ij} - 1 \quad \forall i = 1, 2, \ldots, m - 1; \quad z = i + 1, \ldots, m; \quad j = 1, 2, \ldots, S \]

\[ \sum_{j=1}^{S} x_{ij} = y_{ij}^t + 2y_{ij}^p \quad \forall i = 1, 2, \ldots, m \]

\[ e_{e_{ij}}^t \leq y_{ij}^t + y_{ij}^s - 1 \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad k = s, p; \quad l = s, p \]

\[ e_{e_{ij}}^t \leq y_{ij}^t \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad k = s, p; \quad l = s, p \]

\[ e_{e_{ij}}^t \leq y_{ij}^s \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad k = s, p; \quad l = s, p \]

\[ w_{i_{ij}}^m = e_{e_{ij}}^t + e_{e_{ij}}^p \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m \]

\[ w_{i_{ij}}^t = e_{e_{ij}}^t \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m \]

\[ w_{i_{ij}}^m + w_{i_{ij}}^t + w_{i_{ij}}^p = 1 \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m \]

\[ U_{i_{ij}}^l \geq w_{i_{ij}}^l + x_{ij} - 1 \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad k = i, z; \quad j = 1, \ldots, S; \quad l = s, p, \Omega \]

\[ U_{i_{ij}}^l \leq w_{i_{ij}}^l \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad k = i, z; \quad j = 1, \ldots, S; \quad l = s, p, \Omega \]

\[ U_{i_{ij}}^l \leq x_{ij} \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad k = i, z; \quad j = 1, \ldots, S; \quad l = s, p, \Omega \]

\[ V_{i_{ij}}^{l_a} \geq U_{i_{ij}}^l + C_{i_{ij}} - 1 \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad k = i, z; \quad j = 1, \ldots, S; \quad a = 1, \ldots, S; \quad l = s, p, \Omega \]

\[ V_{i_{ij}}^{l_a} \leq U_{i_{ij}}^l \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad k = i, z; \quad j = 1, \ldots, S; \quad a = 1, \ldots, S; \quad l = s, p, \Omega \]

\[ V_{i_{ij}}^{l_a} \leq C_{i_{ij}} \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad k = i, z; \quad j = 1, \ldots, S; \quad a = 1, \ldots, S; \quad l = s, p, \Omega \]

\[ H_{i_{ij}}^l \geq w_{i_{ij}}^l + C_{i_{ij}} - 1 \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad j = 1, \ldots, S; \quad l = s, p, \Omega \]

\[ H_{i_{ij}}^l \leq w_{i_{ij}}^l \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad j = 1, \ldots, S; \quad l = s, p, \Omega \]

\[ H_{i_{ij}}^l \leq C_{i_{ij}} \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad j = 1, \ldots, S; \quad l = s, p, \Omega \]

\[ \beta_{ij} \geq 0 \quad \forall i = 1, 2, \ldots, m; \quad j = 1, 2, \ldots, S - 1; \quad l = j + 1, j + 2, \ldots, S \]

\[ C_{ij} \geq 0 \quad \forall i = 1, 2, \ldots, m - 1; \quad z = i + 1, \ldots, m; \quad j = 1, 2, \ldots, S \]

\[ e_{e_{ij}}^t \geq 0 \quad \forall i = 1, \ldots, m; \quad z = i + 1, \ldots, m; \quad k = s, p; \quad l = s, p \]
\[ w^e_{iz} \geq 0; \ w^p_{iz} \geq 0; \ w^p_{iz} \geq 0 \quad \forall i = 1,\ldots,m; \ z = i + 1,\ldots,m \]
\[ U^i_{izj} \geq 0 \quad \forall i = 1,\ldots,m; \ z = i + 1,\ldots,m; \ k = i, z; \ j = 1,\ldots,S; \ l = s, p, \Omega \]
\[ V^i_{izj} \geq 0 \quad \forall i = 1,\ldots,m; \ z = i + 1,\ldots,m; k = i, z; j = 1,\ldots,S; a = 1,\ldots,S; l = s, p, \Omega \]
\[ H^i_{izj} \geq 0 \quad \forall i = 1,\ldots,m; \ z = i + 1,\ldots,m; \ j = 1,\ldots,S; l = s, p, \Omega \]

**A.4 Complexity of the Effort Minimization Problem**

**Theorem 1** The pair programming version of the effort minimization problem is strongly NP-hard.

The main idea behind the proof is easy to summarize: an arbitrary instance of the classical bin-packing problem (Garey and Johnson 1979) can be polynomially transformed to the pair programming version of the effort minimization problem. In this construction, an item and a bin in the bin-packing problem correspond, respectively, to a module and a developer pair. Assigning modules to a developer pair corresponds to the notion of packing items in a bin. We choose particular values of the pair formation time such that it is only feasible to construct pairs of the kind: (1, 2), (3, 4), (5, 6), and so on, where the numbers in the parentheses identify developers. All such pairs shown above require the same pair formation time; any other pair (e.g., (2, 3)) is ruled out by setting its pair formation time to exceed the time available for the project. Appropriate choices for the module development times and the project completion time complete the construction of an instance of the pair programming version of the effort minimization problem. For any positive integer \( K \), the decision question “can the project be completed using less or equal to \( K \) pairs?” posed on the constructed instance is equivalent to solving the decision version of the classical bin-packing problem.

Consider the decision problem \( Q_{EP} \) corresponding to the pair programming version of the effort minimization problem.

**Decision Problem (Q_{EP}):** Given the number of groups \( G \), set of modules \( M = \{1,2,\ldots,m\} \), set \( M_g \) for each group \( g \), values \( D^m_{ij} \) for each module \( i \) and each developer pair \( (j,l) \), values \( p_{iz} \) and \( \lambda^p_{iz} \) for each module pair \( (i,z) \), values \( f_{jl} \) for each developer pair \( (j,l) \), values \( \alpha_j \) for each developer \( j \), and values \( S, T, \) and a specified number \( F \), does there exist a valid developer assignment such that the total system development effort is less than or equal to \( F \)?
We now show that the classical bin-packing problem (Garey and Johnson 1979) can be polynomially transformed to $Q_{\text{EP}}$.

**Bin-Packing Problem:** Given a finite set $U = \{u_1, u_2, \ldots, u_m\}$ of items and a size $s(u_i) \in [0, 1]$ for each item $u_i \in U$, does there exist a partition of $U$ into disjoint subsets $U_1, U_2, \ldots, U_k$ such that the sum of the sizes of the items in each $U_j$ is no more than 1, and $k \leq K$, where $K$ is a positive integer?

Given an arbitrary instance of the bin packing problem, we construct an instance of $Q_{\text{EP}}$ as follows. Each module (resp., pair) corresponds to an item $u_i$ (resp., a disjoint subset $U_j$) in the bin packing problem. We have one group (i.e., $G = 1$), $S$ developers, and $m$ modules, where $S$ is any even number such that $S \geq 2m$. $\lambda_{ij}^p = 0 \ \forall i, z$. Let $f$ be any positive real number. The pair formation efforts for different developer pairs are set as follows: $f_{jl} = f \\forall l = j+1, j = 1, 3, 5, \ldots, S-1; \ \ f_{jl} > (2+f) \ \forall l \neq j+1, j = 1, 3, 5, \ldots, S-1; \ \text{and} \ f_{jl} > (2+f) \ \forall l \neq j-1, j = 2, 4, 6, \ldots, S$.

Moreover, we set $T = 1+0.5f$ and $D_{ij}^p = D_i^p = 2s(u_i) \ \forall i, j, l$.

In any valid assignment for this instance of $Q_{\text{EP}}$, the only possible pairs of developers are $(j,l)$ where $j = 1, 3, 5, \ldots, S-1$ and $l = j+1$. For any other developer pair, the time constraint would not be satisfied. Hence, for $F = \sum_{i=1}^m D_i^p + fK$, it is easy to see that the solution of this instance of $Q_{\text{EP}}$ provides a solution to the bin packing problem and vice-versa. Moreover, the bin packing problem is NP-complete in the strong sense (Garey and Johnson 1979). The result follows.

**A.5 Complexity of the Time Minimization Problem**

**Theorem 2** Both solo and pair versions of the time minimization problem are strongly NP-hard.

We show that an arbitrary instance of the multi-processor scheduling problem (Garey and Johnson 1979) can be polynomially transformed to an instance of the time minimization problem. In this transformation, a task and a processor of the multi-processor scheduling problem correspond to a module and a developer in the solo version (or, a developer pair in the pair version) of the time minimization problem, respectively. For a positive integer $D$, the decision question “can the project be completed in less than or equal to $D$ units of time?” posed on the constructed instance of the time minimization problem is equivalent to solving the decision version of the multi-processor scheduling problem.
Consider the following decision problem $Q_{TS}$ corresponding to the solo version of the time minimization problem.

**Decision Problem ($Q_{TS}$):** Given the number of groups $G$, set of modules $M = \{1, 2, \ldots, m\}$, set $M_g$ for each group $g$, a value $D^i_j$ for each module $i$ and each developer $j$, values of $p_{iz}$ and $\lambda^i_z$ for each module pair $(i, z)$, values of $\alpha^i_j$ for each developer $j$, and values of $S$, $B$, and some specified number $F'$, does there exist a valid developer assignment such that the total time needed to complete the project is less than or equal to $F'$?

We now show that the following multi-processor scheduling problem can be polynomially transformed to $Q_{TS}$.

**Multi-Processor Scheduling Problem** (Garey and Johnson 1979): Given a finite set $V = \{v_1, v_2, \ldots, v_m\}$ of tasks, a length $l(v_i)$ for each task $v_i \in V$, and a positive integer $D$, does there exist a partition of $V$ into disjoint subsets $V_1, V_2, \ldots, V_S$ such that

$$\max \left\{ \sum_{a \in V_j} l(a) : 1 \leq j \leq S \right\} \leq D?$$

Given an arbitrary instance of the multi-processor scheduling problem, we construct an instance of $Q_{TS}$ as follows. Each module (resp., developer) corresponds to a task $v_i$ (resp., a disjoint subset $V_j$) in the multi-processor scheduling problem. We set $G = 1$, $\lambda^i_z = 0 \ \forall i, z$, $D^i_j = D^i_j = l(v_i) \ \forall i, j$, $B \geq \sum_{i=1}^{m} l(v_i)$, and $F' = D$. It is now easy to see that the solution of this instance of $Q_{TS}$ provides a solution to the multi-processor scheduling problem and vice-versa. Moreover, the multi-processor scheduling problem is NP-complete in the strong sense for an arbitrary $S$ (Garey and Johnson 1979). The result follows for the solo version.

The proof is similar for the pair version of the time minimization problem.

**A.6 DETAILS OF THE GENETIC ALGORITHM (GA)**

Drezner (2003) proposed several variants of the GA for solving the Quadratic Assignment Problem. We devise GAs for both the effort and the time minimization problems based on a variant that was shown to be both effective and efficient. We refer the reader to Goldberg (1989) and Drezner (2003) for a general introduction to GAs. Here, we summarize the main ideas in the context of our problem.
It is important to note that both the effort and the time minimization problems involve inequality constraints (the time constraint and the effort budget constraint, respectively). One way of handling such constraints in GAs is to use penalties. In this approach, an infeasible solution is avoided by degrading its objective function value proportional to the degree of violation in the constraints (see Goldberg 1989 for details). A constrained problem is thus transformed to an unconstrained problem by associating a penalty for each constraint violation.

Before summarizing our implementation, we describe two procedures: (a) The Cohesive Merging Procedure and (b) The Post-Merging Procedure with Tabu Search. Two key operators of a GA are crossover and mutation. The crossover operator merges two solutions (called “parents”) to produce a new solution. The idea behind this operator is that the new solution may be better than both of the parents if it takes the best characteristics from each parent. In our implementation, this idea is operationalized using the Cohesive Merging Procedure. The mutation operator alters one or more values in a solution to generate a new solution; with such changes, the GA may be able to arrive at a solution that otherwise may not be reachable using the crossover operator. This operator is implemented in our GA using the Post-Merging Procedure with Tabu Search.

**The Cohesive Merging Procedure**

**Figure A.1. An Example to Illustrate the Calculation of Distances**

The Cohesive Merging Procedure is similar to the merging procedure used in Drezner and Salhi (2002) for solving network design problems. Here, we first calculate the distance of each module to all other modules. The distance between a pair of modules is the number of links in the
shortest path between the two modules. To illustrate, Figure A.1 shows the distances of all the modules from Module A.

Next, we randomly select a pair of solutions (i.e., parents) and merge them to produce a new solution as follows. The parent with a smaller value of the objective function is defined as the first parent. If the two parents tie in the value of the objective function, one of them is arbitrarily defined as the first parent. We then generate $m$ new solutions by executing the following for each module $i$ ($i = 1, 2, \ldots, m$).

1. The median distance of module $i$ to all other modules is computed.
2. A module that is closer than the median to module $i$ is assigned the developer(s) from the first parent.
3. All other modules are assigned developer(s) from the second parent.

Finally, we compare the objective function values of the $m$ new solutions and select the one with the smallest objective function value as the new solution.

The Post-Merging Procedure (PMP) with Tabu Search

The PMP is designed to improve the quality of a solution. We begin with a well-known descent heuristic (Armour and Buffa 1963), where we check the objective function values for all pairwise exchanges of developer assignments between the modules in the selected solution. If an improving exchange is found, the best improving exchange is executed. This process is repeated until no improving exchange is found.

The steps of the PMP with Tabu Search are as follows.

1. Select a solution and perform the descent heuristic. Suppose the number of iterations in the descent heuristic is $h$. The terminal solution of the descent heuristic is defined as the current solution as well as the best-known solution.
2. Repeat the following max[$2h$, 50] times:
   (a) Check all pairwise exchanges of developer assignments between the modules in the current solution.
   (b) If a solution better than the best-known solution is found, the best improving exchange is performed, the tabu list is emptied, and the next iteration starts.
   (c) If no solution (from a pairwise exchange) is better than the best-known solution, the best exchange that is not in the tabu list is performed. The two exchanged
modules are added to the tabu list. If the list exceeds a threshold of 20 modules, the two modules with the largest tenure in the list are removed and the next iteration starts.

We now present the details of our implementation; the population size is $P$ and the number of generations is $G$ (see Goldberg 1989 for details).

**The Genetic Algorithm**

1. $P$ solutions are generated randomly and the PMP with Tabu Search is applied to each solution to obtain a starting population.
2. Two members of the population are randomly selected to be parents. The quality of a population member does not affect the probability that it is selected.
3. The two parents are merged using the Cohesive Merging Procedure to produce a new solution.
4. A PMP with Tabu Search is applied to the new solution to create an offspring.
5. If the offspring has a smaller (i.e., better) objective function value than that of the worst member of the population, then it is compared with all existing population members.
   - If the offspring is not identical to an existing population member, the offspring replaces the worst member of the population.
   - The offspring is ignored if it is identical to an existing population member.
6. Repeat Steps 2–5 $G$ times.
7. The best member of the population after $G$ generations is selected as the solution.

**Appendix B: Pair Programming for the Effort Minimization Problem**

From Section 3.1, we have

$$\text{Pair Programming Development Effort} \ (PPDE(S)) = m\sigma d + S f / 2 + (1/2 - 2\alpha / S)mrI$$
$$\text{Pair Programming Development Time} \ (PPDT(S)) = m\sigma d / S + f / 2 + (1/2 - 2\alpha / S)mrI$$

We first observe that $PPDE(S)$ is increasing in $S$. Hence, the optimal value of $S$ should be chosen as the lowest value of $S$ that satisfies the time constraint: $T \leq PPDT(S)$, with the requirement that
there must be at least one pair. Let \( s = S/2 \) denote the number of disjoint pairs. The Pair Programming Development Time can then be rewritten as:

\[
PPDT(s) = 0.5m\sigma d/s + 0.5f + (0.5/s)(0.5 - \alpha/s)mrl
\]

Solving the time constraint above with the equality sign, we get:

\[
s^* = \frac{-m(rI + 2\sigma d) \pm \sqrt{(mrl + 2m\sigma d)^2 + 16(\alpha mrl)(f - 2T)}}{4(f - 2T)}
\]

We first note that the average time required for one pair to be formed, \( f/2 \), must be less than the available time, \( T \). Hence, \( f - 2T < 0 \). If there are 2 valid roots, then we must have

\[
\frac{-m(rI + 2\sigma d) \pm \sqrt{(mrl + 2m\sigma d)^2 + 16(\alpha mrl)(f - 2T)}}{4(f - 2T)} \geq 1
\]

Consider the inequality

\[
\frac{-m(rI + 2\sigma d) + \sqrt{(mrl + 2m\sigma d)^2 + 16(\alpha mrl)(f - 2T)}}{4(f - 2T)} \geq 1 \tag{A}
\]

The above inequality implies that

\[
\sqrt{(mrl + 2m\sigma d)^2 + 16(\alpha mrl)(f - 2T)} \leq 4(f - 2T) + m(rI + 2\sigma d)
\]

Squaring both sides,

\[
(mrl + 2m\sigma d)^2 + 16(\alpha mrl)(f - 2T) \leq 16(f - 2T)^2 + (mrl + 2m\sigma d)^2 + 8(f - 2T)(mrl + 2m\sigma d)
\]

\[
\Rightarrow \quad 16(\alpha mrl)(f - 2T) \leq 16(f - 2T)^2 + 8(f - 2T)(mrl + 2m\sigma d)
\]

\[
\Rightarrow \quad 2(\alpha mrl) \geq 2(f - 2T) + (mrl + 2m\sigma d)
\]

\[
\Rightarrow \quad 0.5f + 0.5m\sigma d + 0.25(mrl)(1 - 2\alpha) \leq T
\]

The left hand side of the last inequality above represents the pair programming development time taken by one pair of developers to develop the entire system. If the available time is greater than the one-pair development time, clearly, only one pair should be used to minimize the development effort. If, however, the available time is less than the one-pair time, the inequality in (A) can never be satisfied. Hence the optimal solution is given by the (larger) root, where the discriminant is used with a negative sign. The ceiling function guarantees that an integer value of \( s \) will be used.
Appendix C: Solo Programming for the Effort Minimization Problem

The proof is on similar lines to the one in Appendix B.

Solo Programming Development Effort = \( \text{SPDE}(S) = md + mrI(S - \alpha)/2S \)

Solo Programming Development Time = \( \text{SPDT}(S) = md/S + mrI(S - \alpha)/2S^2 \)

Once again, we note that the effort is increasing in \( S \) and obtain:

\[
S^* = \frac{m(2d + rI) \pm \sqrt{(m(2d + rI))^2 - 8T(\alpha mrI)}}{4T} \geq 1
\]

Consider,

\[
\frac{m(2d + rI) - \sqrt{(m(2d + rI))^2 - 8T(\alpha mrI)}}{4T} \geq 1
\]

\[\Rightarrow \quad md + 0.5(mrI)(1 - \alpha) \leq T
\]

The left hand side of the above inequality is the time taken by one developer to develop the entire system. If the available time is greater than the one-developer time, then the optimal solution is to use one developer. Otherwise, the optimal solution is given by:

\[
S^* = \left[ \frac{m(2d + rI) + \sqrt{(m(2d + rI))^2 - 8T(\alpha mrI)}}{4T} \right]
\]

Appendix D: Dominance of the Pure Approach over the Mixed Approach

D.1 Effort Minimization In The Absence Of A Time Constraint

Consider an arbitrary assignment under the mixed approach. Let \( M = \{1, 2, \ldots, m\} \) represent the set of all modules. Let \( W \subseteq M \) be the set of modules that have each been assigned a pair of
developers. Thus, modules in $M \setminus W$ are assigned a single developer. Let $|W|/|M|=|W|/m=\rho$. Since there is no time constraint, it follows that any effort-minimizing assignment of developers to modules satisfies the following: (1) all the modules in $S$ are assigned the same developer pair, say $(a,b)$, and (2) all the modules in $M \setminus W$ are assigned developer $a$ (or, alternatively, all of them are assigned to developer $b$). Given the above assignment, the value of the effort, $E^\alpha$, can be easily computed as

$$E^\alpha = \rho m \sigma d + (1-\rho)md + f + \frac{\rho^2 mr}{2}(1-2\alpha)I + \left(\frac{mr}{2} - \frac{\rho^2 mr}{2}\right)(1-\alpha)I$$

Now, we consider two assignments under the pure approach. In an optimal pair programming assignment, all the modules in $M$ are assigned to a single pair, say $(a,b)$. For this assignment, the effort is

$$E_{\text{min}}^p = m \sigma d + f + \frac{mr}{2}(1-2\alpha)I$$

In an optimal solo programming assignment, all modules in $M$ are assigned to a single programmer, say $a$. The effort for this assignment is

$$E_{\text{min}}^s = md + \frac{mr}{2}(1-\alpha)I$$

$$E^\alpha = \rho m \sigma d + (1-\rho)md + f + \frac{\rho^2 mr}{2}(1-2\alpha)I + \left(\frac{mr}{2} - \frac{\rho^2 mr}{2}\right)(1-\alpha)I$$

$$= \rho m \sigma d + (1-\rho)md + f + \frac{mr}{2}(1-\alpha)I - \frac{\rho^2 mr}{2}\alpha I$$

$$\geq \rho m \sigma d + (1-\rho)md + \rho f + \frac{mr}{2}(1-\alpha)I - \frac{\rho mr}{2}\alpha I$$

$$= \rho E_{\text{min}}^p + (1-\rho)E_{\text{min}}^s$$

Thus, for $0 \leq \rho \leq 1$, either $E^\alpha \geq E_{\text{min}}^p$ or $E^\alpha \geq E_{\text{min}}^s$. Note that for $0 < \rho < 1$, $E^\alpha > E_{\text{min}}^p$ or $E^\alpha > E_{\text{min}}^s$. That is, the effort for any strictly mixed assignment (i.e., one with $W \neq \phi$ and $W \neq M$) is strictly greater than that of either the optimal pair programming effort or the optimal solo programming effort. This completes the proof of the dominance of a pure approach over a proper mixed approach.

Finally, it is easy to characterize the situation when a particular pure approach dominates the other. Specifically, $E_{\text{min}}^s \leq E_{\text{min}}^p$ if and only if $mr\alpha I \leq 2md(\sigma - 1) + 2f$. ■
Appendix E: Numerical Experiments

E.1 Artificial System Generation

The system consists of 10 groups (or iterations), namely, invoice enquiry, display contact, payment details, etc. Each group consists of 8-14 modules (tasks) and there are a total of 100 modules in the system. For example, the group invoice enquiry consists of invoice section, display invoice, reimbursement, etc. (names disguised to protect confidentiality). A module may be required by more than one group; for example, the dispute dues module is required by the invoice enquiry, payment details, customer model, and portfolio groups. The structure of the billing system reveals that modules have a relatively higher degree of functional dependence (i.e., there are more links) with other modules in the same group, as opposed to the dependence with modules in other groups. In this project a group typically has approximately 15-20 internal links and approximately 3-7 external links. In all, there are a total of approximately 250 links in the billing system.
E.2 Numerical Factor Values

Drawing from the details of this system, we generated artificial systems where the “within group” density of links was much higher than the “across group” density. Another feature of the billing system was that the modules in the system required different development efforts. To capture this, we generated systems with modules that corresponded to three effort levels: complex, average and simple with development efforts (for solo programming) normalized to 1, 0.5, and 0.25 person-weeks, respectively. To permit some variation in the distribution of development effort, for a system with $m$ modules, the number of modules with each effort level was chosen randomly from a uniform distribution with mean equal to $m/3$ and coefficient of variation equal to $1/\sqrt{3}$. Following recent empirical studies (Kofax Image products White Paper 2001), the average integration effort for a link was chosen to be directly proportional to the average development effort of the two connected modules.

The number of modules ($m$) in the system was chosen to be 40 and 70 corresponding to small and large systems, respectively. These modules were spread across 6 groups. The mean number of modules per group was $m/6$ and the coefficient of variation of the number of modules per group was $\sqrt{3}/10$.

For a system with $m$ modules and $\varphi$ links, the mean number of connections per module is $(2\varphi/m)$ and the coefficient of variation is $\sqrt{(m^2 - 2\varphi)/(2\varphi m)}$. The link density $L$, given by $2\varphi/(m(m-1))$, is set to 0.05 (low density) and to 0.1 (high density). In addition, we checked to make sure that connected systems were generated, i.e., no subset of modules was completely isolated from the rest of the system. Finally, in keeping with our goal of generating realistic systems, we ensured that the number of links within a group was much higher than the number of links across any two groups.

The mean knowledge sharing coefficient ($\alpha$) is used in our model to quantitatively capture the knowledge sharing advantage provided during integration when modules being integrated share common developers. We chose the mean values of this factor at two levels: 0.02 and 0.3 respectively for low and high knowledge sharing. To allow the knowledge sharing coefficient to change with the level of expertise of the developer being shared, we use a multiplier of 1.2 (times the mean value) when an expert developer is shared, a multiplier of 1 when an average developer is shared, and a multiplier of 0.8 when a novice developer is shared.
The values of the pair development overhead (σ) are chosen based upon statistical experiments involving pair programming (Cockburn and Williams 2000), where it was discovered that two people worked almost twice as fast (about 10-15% more person-hours) when compared to individuals. Based on these observations, the values of the pair development overhead were chosen to be 1.02 and 1.3 respectively for low overhead and high overhead.

The base integration effort for a link (i.e., assuming no commonality) was chosen to be dependent on the effort needed to develop the two modules associated with the link and the development approach used to develop these modules. This effort was chosen such that the efforts of the different development approaches were in the ratio of 4:3:2 corresponding to solo-solo, solo-pair, and pair-pair, respectively.

The values of the mean pair formation effort (f) in our experiment are drawn from observations indicating that a jelling time between 1 to 40 hours is required for pair formation (Erdogmus and Williams 2003). In other industry reports, this adjustment period has historically taken anywhere from hours to days, depending upon the individuals (Williams et al. 2000). Given the high degree of variability reported, we consider a wide range of values for the pair formation effort. The mean pair formation effort was chosen to be at two levels, high and low. Corresponding to the effort needed to develop a module of average complexity (0.5 person-weeks), we set the high level of the mean pair formation effort to 0.5 person-weeks and the low level to a fifth of this value, or 0.1 person-weeks. Next, depending on the expertise levels of the developers in the pair, the actual pair formation effort for a pair was set equal to a multiplier times the mean pair formation effort. This multiplier was varied in steps of 0.1, starting at 0.8 (corresponding to the best case where both members of the pair being formed were expert developers) and ending at 1.2 (corresponding to the case where both members of the pair being formed were novice developers). The multiplier for an expert-novice pair was set equal to the multiplier for an average-average pair; both these values were set equal to 1.

Project Deadline (T): the project deadline was set for every problem instance by first solving a time minimization model with no budget constraint. The solution to this model then provided a lower bound to the time that must be available for a particular problem instance to be feasible. For each problem instance, we first computed $T_s$ and $T_p$ as the minimum feasible time needed to complete the project using solo and pair programming respectively. Next, we set $T^* = \max (T_s, T_p)$. The
The deadline for the problem was then chosen as $T = k \ T^*$, where $k = (1.3 \text{ and } 2.1)$ to represent projects with tight and relaxed deadlines respectively.

The level of expertise in a project team was measured by the number of expert developers in the team. The number of average developers was always chosen to be equal to 3 and the number of experts ($n_e$) was chosen to be either 1 (low level of expertise) or 3 (high level of expertise). Finally, the number of novice developers was chosen to be equal to $4 - n_e$.

### E.3 Regression Experiment

#### Table E.1
**Regression Estimates**

<table>
<thead>
<tr>
<th>Term</th>
<th>Estimate</th>
<th>Significance</th>
<th>VIF</th>
<th>Tolerance</th>
<th>Condition Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n_e$</td>
<td>-2.599</td>
<td>&lt;0.001</td>
<td>3.99</td>
<td>0.25</td>
<td>1.00</td>
</tr>
<tr>
<td>$m$</td>
<td>0.898</td>
<td>&lt;0.001</td>
<td>4.35</td>
<td>0.23</td>
<td>6.00</td>
</tr>
<tr>
<td>$Xf$</td>
<td>2.750</td>
<td>0.0024</td>
<td>1.91</td>
<td>0.52</td>
<td>1.43</td>
</tr>
<tr>
<td>$XL$</td>
<td>125.51</td>
<td>&lt;0.001</td>
<td>5.53</td>
<td>0.18</td>
<td>1.60</td>
</tr>
<tr>
<td>$X\alpha$</td>
<td>-34.524</td>
<td>&lt;0.001</td>
<td>3.99</td>
<td>0.25</td>
<td>1.78</td>
</tr>
<tr>
<td>$XT$</td>
<td>-10.367</td>
<td>&lt;0.001</td>
<td>3.57</td>
<td>0.28</td>
<td>1.97</td>
</tr>
<tr>
<td>$R_{\alpha m}$</td>
<td>-0.339</td>
<td>0.0291</td>
<td>1.00</td>
<td>1.00</td>
<td>1.98</td>
</tr>
<tr>
<td>$R_{X\alpha \sigma}$</td>
<td>17.988</td>
<td>0.0009</td>
<td>1.84</td>
<td>0.54</td>
<td>2.25</td>
</tr>
<tr>
<td>$R_{XfT}$</td>
<td>-12.522</td>
<td>0.006</td>
<td>2.06</td>
<td>0.48</td>
<td>2.73</td>
</tr>
<tr>
<td>$R_{X\alpha m}$</td>
<td>0.193</td>
<td>&lt;0.001</td>
<td>1.46</td>
<td>0.68</td>
<td>3.10</td>
</tr>
<tr>
<td>$R_{X\alpha \sigma T}$</td>
<td>6.839</td>
<td>&lt;0.001</td>
<td>2.87</td>
<td>0.35</td>
<td>3.60</td>
</tr>
<tr>
<td>$R_mL$</td>
<td>6.411</td>
<td>&lt;0.001</td>
<td>1.07</td>
<td>0.93</td>
<td>4.67</td>
</tr>
<tr>
<td>$R_{XmL}$</td>
<td>-4.560</td>
<td>&lt;0.001</td>
<td>1.73</td>
<td>0.58</td>
<td>5.37</td>
</tr>
</tbody>
</table>

#### Table E.2
**Orthogonalized Interaction Variables**

- $R_{\alpha m} = \alpha m + 8.8-55\alpha-0.16m$
- $R_{X\alpha \sigma} = X\alpha \sigma + 0.252 - 0.16X - 0.58\alpha - 0.16\sigma$
- $R_{XfT} = XfT + 0.075 - 0.5f - 0.15T$
- $R_{X\alpha m} = X\alpha m + 86.9 - 0.55X - 0.55\alpha - 0.58m$
- $R_{X\alpha \sigma T} = X\alpha \sigma T + 0.79 - 0.5X - 0.5\alpha - 0.58T$
- $R_mL = mL + 4.522 - 0.077m - 55.54L$
- $R_{XmL} = XmL + 4.329 - 4.135X - 0.038m - 28.77L$
TABLE E.3
SUMMARY OF EFFECTS

<table>
<thead>
<tr>
<th>INCREASE IN PARAMETER</th>
<th>IMPACT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PAIR</td>
</tr>
<tr>
<td>NUMBER OF MODULES</td>
<td>+</td>
</tr>
<tr>
<td>LINK DENSITY</td>
<td>+</td>
</tr>
<tr>
<td>KNOWLEDGE-SHARING COEFFICIENT</td>
<td>−</td>
</tr>
<tr>
<td>PAIR FORMATION EFFORT</td>
<td>+</td>
</tr>
<tr>
<td>PAIR DEVELOPMENT OVERHEAD</td>
<td>+</td>
</tr>
<tr>
<td>DEADLINE</td>
<td>−</td>
</tr>
<tr>
<td>NUMBER OF EXPERTS</td>
<td>−</td>
</tr>
</tbody>
</table>

+, −, 0: Increasing, Decreasing, and No impact, respectively.

+, ++, − −: More positive or More negative relative impact, respectively.

TABLE E.4
ERROR TERM DIAGNOSTICS

<table>
<thead>
<tr>
<th>TEST FOR LOCATION $\mu_0=0$</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>STUDENT’S T-TEST</td>
<td>0.3162</td>
</tr>
<tr>
<td>SIGNED</td>
<td>0.2880</td>
</tr>
<tr>
<td>SIGNED RANK</td>
<td>0.2059</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TEST FOR NORMALITY</th>
<th>P-VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>KOLMOGOROV-SMIRNOV</td>
<td>0.0715</td>
</tr>
</tbody>
</table>

References


