Time-line based model for software project scheduling with genetic algorithms

Carl K. Chang, Hsin-yi Jiang, Yu Di, Dan Zhu, Yujia Ge
Information and Software Technology(IST), 2008

2010. 3. 9
Presented by Dongwon Kang

ⓒ KAIST SE LAB 2008
Contents

- Introduction
- Overview of the previous model
- Specification of the new model
- Experiments
- Related work
- Conclusion
- Discussion
Introduction

- Task assignments are crucial to software development
  - Quality of the assignments greatly influences the projects
- Most existing project management tools only provide passive project tracking and reporting aids
- This research proposes GA-based project scheduling approach
  - By extending their previous approach [Annals of SE, 2001]
    - Enabling more fine-grained allocation using time-lines
      - By breaking down development tasks into smaller time-sliced activities
    - Considering dynamics of software projects
      - Effect of learning in project execution (experience and training)
Overview of the previous model (1/2)

- Representation of a project: a task precedence graph
  - Example)

- Representation of an employee: skills and salary

- Objective function
  - $F = validity \times \frac{W_1}{Overload} + \frac{W_2}{Cost} + \frac{W_3}{Time}$
    - Validity is 1 if the assignment is valid, otherwise 0.
      - Decided by skill matching, precedence of tasks, covering all tasks
Overview of the previous model (2/2)

- Problems of the previous approach
  - Execution of tasks in an atomic way
    - Interruption of task execution is not allowed
      - A new critical task can be executed during execution of other tasks
  - Assignment strategy
    - It is not realistic to assume that adding more employees to a task always produce better schedule
      - It increases communication and coordination cost
  - Skills and experience
    - Different levels of skills and experience are not supported
  - Different scales for objectives
    - Difficulty in setting weight for three objectives
Introduction of time-line
- To enable addition and removal of employees during task execution
  - By breaking down development tasks into smaller time-sliced activities

Revision of the employee model
- Skill model
  - The level of skill is measured as a number between 0 and 5
  - A skill can be obtained during a project through training
    - Proficiency \( p = \min\{p_0 + u \times \text{trainingTime}, 5\} \)
      - \( p_0 \): proficiency at the beginning, \( u \): learning speed
- Experience model
  - Considered effect from on-the-job training related to tasks
    - Experience \( e = \min\{e_0 + u \times \text{taskTime}, 5\} \)
- Productivity of an employee for a task \( t \):
  \[
  \frac{e_t}{5} + \left( \prod_{s \in S(t)} \frac{p_s}{5} \right)^{2/5}
  \]
Specification of the new model (2/6)

- Assignment scheme in time-line based model

- Employee can partially participate in each time slot (0%, 25%, 50%, 75%, 100%)
Revision of the employee model (Cont’d)

- Compensation model according to overtime working
  - \( \text{Payment} = \begin{cases} 
  \text{Salary}_{\text{normal}} \times h\% & (0 \leq h \leq 100) \\
  \text{Salary}_{\text{normal}} \times 100\% + \text{Salary}_{\text{overtime}} \times (h-100)\% & (100 < h \leq h_{\text{max}}) \\
  \infty & (h_{\text{max}} < h) 
  \end{cases} \)

- Purpose: integrating different objectives with one measure - cost
  - This approach also assumes that schedule can be considered with cost
Specification of the new model (4/6)

- Revision of the task model
  - Deadline issues
    - By imposing penalty when a task exceeds the deadline
      \[
      \text{penalty} = \begin{cases} 
      0 & M \leq D_{\text{soft}} \\
      P \times (M - D_{\text{soft}}) & D_{\text{soft}} \leq M \leq D_{\text{hard}} \\
      \infty & M > D_{\text{hard}} 
      \end{cases}
      \]
    - M is finished time of a task
  - Maximum headcount
    - Setting the limit of the number of employees who can work together effectively for a given task
      - Using COCOMO equation related to the ordinary number of developers
        \[
        \frac{PM}{TDEV} \approx \frac{1}{3} PM^{0.672}
        \]
      - In this study, the team size is simply doubled
        \[
        \text{max} (1, \text{round}(2/3 \times PM^{0.672}))
        \]
Procedure to decide the fitness of an assignment

- After creating a new solution by mutation or crossover,
Specification of the new model (6/6)

- Procedure to decide the fitness of an assignment (cont’d)
  - Heuristics to avoid infeasible assignments
    - 1. Eliminate employees assigned to finished tasks
    - 2. Eliminate employees assigned to untimely tasks using task dependencies
    - 3. Eliminate unsuitable assignments according to skill matching
    - 4. Eliminate unavailable employees
    - 5. Eliminate contractors from training tasks
    - 6. Limit employee training
      - Employees with proficiency more than 4.5 are not supposed to be trained
    - 7. Adjust workloads of overly participating employees
      - If they exceed the maximum value of overtime (invalid)
    - 8. Adjust overstaffed tasks according to maximum headcount
Experiments (1/2)

Experimental design

- Evaluate the fitness by comparing experts’ assignment
  - An artificial project environment was created
    - Consists of 15 tasks, 10 employees, and 5 skills
  - Two experts with experience of more than 20-years joined this experiment
    - Expert 1 tried to optimize schedule
    - Expert 2 tried to optimize both cost and schedule
Experiments (2/2)

Result

<table>
<thead>
<tr>
<th></th>
<th>Cost (Fitness)</th>
<th>Time spent</th>
</tr>
</thead>
<tbody>
<tr>
<td>This approach</td>
<td>527104 (best:385818)</td>
<td>33 min. on average</td>
</tr>
<tr>
<td>Expert 1</td>
<td>546353</td>
<td>3 hours* on average</td>
</tr>
<tr>
<td>Expert 2</td>
<td>559920</td>
<td>(including time to understand the project)</td>
</tr>
</tbody>
</table>

- Allocation from this approach showed better fitness than those from the two experts
  - Best cost can be treated as a lower bound of project cost, and the assignment can be an useful reference to managers
### Related work

<table>
<thead>
<tr>
<th></th>
<th>This paper</th>
<th>Duggan’s (2004)</th>
<th>Ngo-The’s (2009)</th>
<th>My approach</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Goal</strong></td>
<td>Minimize cost of a software project</td>
<td>Optimize schedule and quality in</td>
<td>Maximize business value in release</td>
<td>Generate practical allocation with considering</td>
</tr>
<tr>
<td></td>
<td></td>
<td>implementation</td>
<td>planning</td>
<td>constraints</td>
</tr>
<tr>
<td><strong>Unit of allocation</strong></td>
<td>A time-sliced task</td>
<td>A package in Impl. phase</td>
<td>A time-sliced phase for each module</td>
<td>A phase for each module</td>
</tr>
<tr>
<td><strong>Optimization</strong></td>
<td>Genetic algorithm (GA)</td>
<td>GA</td>
<td>ILP (integer linear programming) + GA</td>
<td>Simulated annealing</td>
</tr>
<tr>
<td><strong>Algorithm</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Productivity</strong></td>
<td>O (by considering skills for tasks)</td>
<td>O</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td><strong>for modules</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Productivity</strong></td>
<td>N/A</td>
<td></td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td><strong>for phases</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Consideration</strong></td>
<td>Effect of learning, Team size</td>
<td>Team size</td>
<td>X</td>
<td>Continuity of allocation, Sharing developers</td>
</tr>
<tr>
<td><strong>of software</strong></td>
<td></td>
<td></td>
<td></td>
<td>between teams, Team size, Expert allocation</td>
</tr>
<tr>
<td><strong>project</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>characteristics</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Guideline</strong></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td><strong>for estimating</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>productivity</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Supporting</strong></td>
<td>O</td>
<td>O</td>
<td>X</td>
<td>O</td>
</tr>
<tr>
<td><strong>co-work</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Validation</strong></td>
<td>Comparing with expert allocation</td>
<td>Example illustration</td>
<td>Showing results from random samples</td>
<td>Using industry project data</td>
</tr>
</tbody>
</table>
Conclusion

**Contribution**
- Proposed GA-based scheduling approach
  - Supported more fine-grained optimization of allocation using time-lines
  - Incorporated dynamic nature of software projects
    - Considered training and experience gained in the project
- Can guide project managers in scheduling software projects

**Future work**
- Improving system dynamics modeling for software projects
- Improving representation of skills, employees, and tasks
- Dealing with various relationships of task dependency
  - End-start, start-start,…
Discussion

❖ Pros
  ▪ Considering effect of learning in scheduling software projects
  ▪ Enabling fine-grained optimization using time-lines

❖ Cons
  ▪ Unrealistic situation may occur from using this approach
    • “Working -> training -> working->…”
    • Continuity of allocation is not supported
      – Employees can be frequently exchanged among tasks
        » Increase overhead due to understanding tasks and context changes
  ▪ No consideration of skill improvement from on-the-job training
  ▪ In experiment,
    • Unfair evaluation in the comparison with expert results
      – Comparison is conducted using cost, but experts focused on the schedule
    • No validation of practicality of allocation results
      – In spite of expert participation in the experiment