Assigning tasks in 24-h software development project

Pankaj Jalote, Gourav Jain

Journal of Systems and Software (JSS ‘06)

2007. 8. 8

HyeonJeong Kim
Contents

- Introduction
- Overall approach
  - Project execution model
  - Resource model
  - Task scheduling algorithm
- Experimental evaluation
- Conclusion
- Discussion
Introduction (1/2)

- 24-h software development project
  - Develop software with different time-zones and locations
Strength and weakness of 24-h software development

- **Strength**
  - Reduce the schedule to develop software

- **Weakness**
  - Complicate task scheduling and project management

**Goal of this paper**
- Provide a task scheduling method to reduce the overall time for a project in 24-h software development
Overall approach

1. Define project execution model
2. Build resource model
3. Run task scheduling algorithm
4. Compare the results with single-site one

Project execution model

Resource model:
- R1 = {m1, m2}
- R2 = {n1, n2, n3}

Task scheduling

Scheduling, resource utilization result
Software process is defined as set of activities or tasks
- Task
  - Definition – Smallest unit of work (node)
  - Constraints – operational, skill constraints, resource constraints

Task graph

Real process

Operational constraints

Skill constraints

Man days

Entry node

Exit node

Resource constraints

Man days

Entry node

Exit node

Man days
Weight of the tasks

- Largest path length (execution time) from the node to the exit node

\[ \text{weight}(\tau) = \max_k \sum_{i \in \phi_k} \text{time}(i) \]

- Higher weight task has higher priority
Resource model (1/2)

- **Assumption**
  - Three timeslots in a day under 24-h software development
  - Each timeslot has its own resource-set

![Diagram](image)

- Timeslot 1
- Timeslot 2
- Timeslot 3

<table>
<thead>
<tr>
<th>Time unit – one day</th>
</tr>
</thead>
<tbody>
<tr>
<td>0h</td>
</tr>
<tr>
<td>8h</td>
</tr>
<tr>
<td>16h</td>
</tr>
<tr>
<td>24h</td>
</tr>
</tbody>
</table>

- No overlap between R1 and R2

- \( R_1 = \{r_1, r_2, \ldots, r_m\} \)
- \( R_2 = \{r_{m+1}, r_{m+2}, \ldots, r_n\} \)
- \( R_3 = \{r_{n+1}, r_{n+2}, \ldots, r_o\} \)
## Resource model (2/2)

### Resource table
- Resource set, skill sets, work period

<table>
<thead>
<tr>
<th>Resource</th>
<th>Resource set</th>
<th>Skill sets</th>
<th>Work period</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_m$</td>
<td>$R_1$</td>
<td>$S_2$</td>
<td>0~8h</td>
</tr>
<tr>
<td>$r_n$</td>
<td>$R_2$</td>
<td>$S_1$</td>
<td>8~16h</td>
</tr>
<tr>
<td>$r_0$</td>
<td>$R_3$</td>
<td>–</td>
<td>16~24h</td>
</tr>
</tbody>
</table>
Task scheduling algorithm (1/3)

- **Initialization**
  - Ready queue Q contains the tasks that are ready to execute
    - Put all the successors of the entry node in the ready queue Q
  
- **Step1**
  - Update the nodes in the ready queue whenever any task complete
    - Put the successor of the complete task with ready state
Step 2

- Divide ready queue Q into two groups
  - Tasks that have no skill and resource constraints – $\tau_q$
  - Tasks that have skill or resource constraints – $\tau_r$
Step 3

- Allocate the resource to the task
  - Higher weight task is executed first
  - $\tau_q$ and $\tau_r$ both exist under following condition
    \[
    weight(\tau_q) - weight(\tau_r) < time(\tau_q)
    \]

Execute task 4 first
Example

Task graph

Resource model

<table>
<thead>
<tr>
<th>Resource</th>
<th>Resource set</th>
<th>Skill sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>l_1</td>
<td>R_1 (0~8h)</td>
<td>S_1</td>
</tr>
<tr>
<td>l_2</td>
<td>R_1 (0~8h)</td>
<td>S_2</td>
</tr>
<tr>
<td>m_1</td>
<td>R_2 (8~16h)</td>
<td>–</td>
</tr>
<tr>
<td>m_2</td>
<td>R_2 (8~16h)</td>
<td>S_2</td>
</tr>
<tr>
<td>n_1</td>
<td>R_3 (16~24h)</td>
<td>S_1</td>
</tr>
</tbody>
</table>
Experimental evaluation (1/3)

- **Experimental design of synthetic projects**

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># of task graphs</td>
<td>100</td>
</tr>
<tr>
<td># of tasks in a graph</td>
<td>5~100</td>
</tr>
<tr>
<td>Execution time of each task</td>
<td>1~20 man–days</td>
</tr>
<tr>
<td>Rate of tasks with skill or resource constraints</td>
<td>10%~70% of tasks</td>
</tr>
<tr>
<td># of resources in one resource-set (p)</td>
<td>3, 5, or more–than 5</td>
</tr>
</tbody>
</table>
Experimental evaluation (2/3)

- Result of synthetic project
  - Compare with a case that all the resource work at single location
  - Schedule reduction (unit: %)
    - Improvement rate compared with single site
    - Resource utilization (unit: %)

<table>
<thead>
<tr>
<th></th>
<th>p=3</th>
<th>p=5</th>
<th>p&gt;5</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;10%</td>
<td>62</td>
<td>38</td>
<td>27</td>
</tr>
<tr>
<td>10-20%</td>
<td>26</td>
<td>48</td>
<td>52</td>
</tr>
<tr>
<td>&gt;20%</td>
<td>12</td>
<td>14</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>p=3</th>
<th>p=5</th>
<th>p&gt;5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single R set</td>
<td>67</td>
<td>61</td>
<td>54</td>
</tr>
<tr>
<td>Three R set</td>
<td>78</td>
<td>68</td>
<td>63</td>
</tr>
</tbody>
</table>

p: # of resource in one resource set
Result of applying task allocation method to two real projects that were executed in single location

<table>
<thead>
<tr>
<th>Phases</th>
<th>Single resource-set</th>
<th>Multiple resource-sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement analysis</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Project management and scheduling</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Screen prototyping</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Functional Spec.</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>Sample application</td>
<td>11</td>
<td>9</td>
</tr>
<tr>
<td>Architecture and database design</td>
<td>12</td>
<td>10</td>
</tr>
<tr>
<td>Detail design</td>
<td>9</td>
<td>7</td>
</tr>
<tr>
<td>Building</td>
<td>16</td>
<td>12</td>
</tr>
<tr>
<td>Unit testing</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>System testing and deployment</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>Overall</td>
<td>88</td>
<td>72</td>
</tr>
</tbody>
</table>

Not specified
Conclusion

● Contribution
  ● Provide a heuristic for scheduling based on the critical path method
    ● Schedule reduce in 24-h model
    ● Resource utilization improves
  ● Case study on some synthetic projects and two real projects
    ● Prove schedule reduction compared with single-site cases
Discussion (1/2)

- Contribution
  - Simplify the task allocation problem with skill set

- Critique
  - Strong assumption
    - Ignore communication overhead or management that is major weakness in 24-h software development
  - Case study
    - Need comparison with the case that doesn’t use task allocation algorithm under 24-h software development
Research topic

- Task scheduling problem with resources that have different productivity on specific tasks
  - NPC problem
- Simplified productivity – skill constraints
  - Provide a heuristic for the task scheduling problem with multiple projects in one organization