Overcoming the Scheduling Barriers in Software Project: 
A Revisit of the Relation Between Development Time and Development Effort 

Liguo Yu, Computer Science Department, Indiana University South Bend, South Bend, IN, USA 

ABSTRACT

Scheduling is an important management activity in software projects. The relationship between nominal development time and development effort has been characterized as a power-law function where development time is dependent on development effort. Studies performed in 1980s determined that there is a scheduling barrier for any software project given the amount of effort to complete the project. Specifically, the studies found out that it is extremely difficult to reduce the development time lowering than a threshold calculated based on the nominal development time. Scheduling barriers are largely due to the complexity of interactions and communications among developers. Thirty years later, this paper examines six software projects in order to understand the relationship between development time and development effort in the 21st century. The author’s study suggests that in modern software development, scheduling barriers could be broken through improving the efficiencies of communications, collaborations, and project management. 

KEYWORDS


1. INTRODUCTION

In software projects, development time is one of the most important factors that customers are concerned with. It directly affects whether the product can be deployed or released on time. In modern software industry, development time is even more critical. Due to the marketplace competitions, it is business-critical for a software product to be released on time or earlier. Unfortunately, the reality is that most software products are delivered over scheduled deadline. 

Increasing the size of development team could not guarantee to reduce the development time, because this strategy might increase the communication complexity and collaboration complexity among developers, which in turn could make the product to be delivered even late. Therefore, in software projects, it is a common practice to determine scheduling based on the estimated development effort. Several formulas have been proposed and validated to show the ideal relationship between development time and development effort (Walston & Felix, 1977; Putnam, 1978; Boehm, 1981). Table 1 summarizes some of the commonly used formulas to estimate the time duration ($T_{\text{nominal}}$) of a project based on the estimated development effort ($E$). 

In Boehm’s 1981 publications (Boehm, 1981), he presented a concept called impossible region, which stated that it is impossible to a schedule a software project’s development time to be less than
three-fourths of its nominal development time, given the amount of effort to complete the project. This scheduling barrier has influenced software industry for over 30 years. Boehm’s observation basically has two implications. First, development time cannot be reduced simply through adding more personnel, because this will increase the overhead of communications. Second, because development time is dependent on development effort, if development effort could be reduced, the development time will be reduced. In the past 30 years, research has been focusing on reducing the development effort in order to reduce the development time. Examples of such techniques include software reuse, component based development, automatic and generative software development.

Despite significant achievement has been made in software research and software industry, scheduling barriers that specify the empirical relations between development time and development effort have not been proved to be removable. In this paper, we re-examine the scheduling barrier thirty years later after it was discovered. Our study is performed on six software datasets, which contain 877 projects. The objective of this study is to answer the following two questions: (1) Are scheduling barriers still valid in modern software development; and (2) How to overcome scheduling barriers.

The remainder of the paper is organized as follows. Section 2 presents the background knowledge. Section 3 describes the data source and the data representation. Section 4 presents the analysis and the results. Conclusions appear in Section 5.

### 2. BACKGROUND

Software project scheduling is important for a company to deliver a quality product on time and within budget (Chen & Zhang, 2013; Xiao et al., 2013; Peixoto et al., 2014; Crawford et al., 2014; Seo et al., 2015; Alhumrani & Qureshi, 2016). Software project scheduling depends on the estimation of the development effort, which is represented as the required manpower to develop the software product (Albrecht & Gaffney, 1983; Jorgensen, 1995). Therefore, effort estimation is the basis for project scheduling and staff allocation. Basically, there are two types of effort estimation methods: expert judgment (Park, 1988; Tausworthe, 1981) and algorithmic model (Carmichael, 2006; Aljahdali & Sheta, 2010). Expert judgment could produce biased estimation. Moreover, it is hard to document and justify estimation given by expert judgment. Therefore, algorithmic models are commonly adopted in software industry. In algorithmic models, COCOMO II is considered the most successful approach (Boehm et al., 2000).

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Walston-Felix</td>
<td>$T_{nominal} = 2.5E^{0.35}$</td>
</tr>
<tr>
<td>COCOMO</td>
<td>$T_{nominal} = 2.5E^{0.28}$</td>
</tr>
<tr>
<td>COCOMO 2</td>
<td>$T_{nominal} = 3.0E^{0.33}$</td>
</tr>
<tr>
<td>Putnam</td>
<td>$T_{nominal} = 2.4E^{1/3}$</td>
</tr>
</tbody>
</table>

Table 1. Nominal development time (Van Vliet, 2008)
Benchmarking: A Method to Improve the Entity's Performance and Change Process
www.igi-global.com/chapter/benchmarking/236939?camid=4v1a