MEASURING SOFTWARE ENGINEERING MATURITY: A RASCH CALIBRATION

David Drehmer  
*DePaul University*

Sasa Dekleva  
*DePaul University*

Follow this and additional works at: [http://aisel.aisnet.org/icis1993](http://aisel.aisnet.org/icis1993)

Recommended Citation

[http://aisel.aisnet.org/icis1993/39](http://aisel.aisnet.org/icis1993/39)
MEASURING SOFTWARE ENGINEERING MATURITY: 
A RASCH CALIBRATION

David Drehmer
Sasa Dekleva
College of Commerce
DePaul University

ABSTRACT

Humphrey, Kitson and Kasse have proposed that those who wish to evaluate the risks, productivity and 
quality associated with choosing a software vendor would be advised to examine the maturity level of the 
software processes employed. Humphrey’s model, rooted in the quality control literature, has lacked 
empirical support. An analysis of data collected in an extended software maintenance study has shown 
that the responses to Humphrey’s key practice items fit the Rasch psychometric model. The authors offer 
an alternative growth interpretation based on their Rasch model analysis.

1. INTRODUCTION

The importance of software reliability is well recognized 
but its achievement difficult. Advances in software engi-
neering do not parallel those made in hardware engineering 
and hardware reliability assessment. A civil engineer, for 
example, can mathematically confirm that a bridge design 
will hold a certain weight. Software engineering as a 
discipline has not yet achieved such a level of maturity 
(Schach 1985). There is no known way to prove that a 
sizable software module, much less a whole system, is 
error-free.

Software failures have been blamed for such problems as 
missing invoices, drawbridges opening without warning in 
rush hour, traffic system malfunctions that caused long 
delays, and even fatalities. Schwartz (1991) described an 
incident where a seemingly innocent sequence of keystrokes 
from the operators of a radiation therapy machine provided 
a lethal dose of radiation to the patients. In another exam-
ple, a software flaw changed the radar timing of a Patriot 
missile and caused it to miss an Iraqi Scud. Twenty-eight 
people died and many others were wounded. Three lines of 
computer code out of two million caused Pittsburgh, Los 
Angeles and Washington to experience an electronic com-
munication paralysis (Neumann 1990).

Eliminating these defects is almost impossible. Parnas 
(1985) has argued that systems such as the Strategic De-
fense Initiative, commonly called “Starwars,” are so 
complex that their failure in some critical aspect is assured.

Existing software engineering strategies are unable to cope 
with the complexity of interacting components. If problems 
were detected in field conditions, no time would be avail-
able to provide a reliable fix.

Various philosophies, design, coding, review, testing and 
management techniques have been proposed to improve the 
software engineering process. The use of these techniques 
has been thought to improve software engineering produc-
tivity and software quality. Standards such as ISO 9000, 
IEEE 730.1-1989, and DOD-STD-2167A have been pro-
posed to create a known level of quality assurance (Mar-

At the request of the U. S. Air Force, the Software Engi-
neering Institute (SEI) at Carnegie Mellon University 
developed a Software Capability Evaluation (SCE). Gov-
ernment agencies could then judge how capable companies 
are at developing software (Bollinger and McGowan 1991; 
Humphrey 1989). According to Anthès (1990), companies 
view a favorable software process maturity assessment by 
SEI or its licensees as a competitive advantage, particularly 
when bidding on federal government jobs. SEI has asserted 
that a higher maturity level is associated with lower risk, 
higher productivity and higher quality of the software 
process (Humphrey, Kitson and Kasse 1989).

These ratings not only will have an effect on U. S. Govern-
ment procurement processes, but also will be used by 
private industry as a basis for deciding who should be 
awarded contracts and who should not.
Adapted from Crosby’s (1979) quality management maturity grid, SEI defined a software engineering process maturity model containing five maturity levels labeled initial, repeatable, defined, managed, and optimized (Humphrey 1988, 1989; Humphrey Kitson and Kasse 1989; Kennet and Koenig 1988).

Each process maturity level was thought to represent a distinctive evolutionary plateau. The framework “models the stages that an organization must go through to establish a culture of engineering excellence. Each model stage lays the foundation on which effective practices for the next level are built” (Humphrey and Curtis 1991, p. 45). The five maturity levels provide the top-level of a multiple layered structure of the software engineering capability model (Humphrey 1988; Weber et al 1991) and a guideline for the software process improvement. A description of each of these levels may be seen in Table 1.

A questionnaire was developed to make a preliminary assessment of software engineering maturity level (Humphrey, Kitson and Kasse 1989). Each maturity level was composed of clusters of related practices. These practices were specific policies, procedures, and activities that represented the link between the maturity model and the questionnaire. Thus, the key practices specified indicators used to generate questionnaire items.

The SCE is not without its critics. Bollinger and McGowan (1991) have pointed out that SCE and process maturity ratings are largely unproven and unvalidated. Documentation of the utility of this process maturity framework has often been anecdotal and sometimes questionable (Humphrey 1988). While SEI promotes that this system should be used by those who wish to improve the process of developing and maintaining software and those who wish to evaluate the risk associated with a software vendor (Paulik, Curtis and Chrissis 1991, pp. 41, 42), the analytical system derived by SEI lacks empirical support to validate the hierarchical nature of its maturity classifications. Further, Bollinger and McGowan indicated many SEI assesses complain about being able to respond affirmatively to level 3 (defined) items but are graded at level 2 (repeatable) because they missed two or more level 2 items. If these items were properly calibrated, such a practice should rarely occur.

"Vendors are credited for achieving the spirit of a key practice even if their procedure differs from the precise wording of a question" according to SEI (Humphrey and Curtis 1991, p. 45). The practice of modifying answers to the questions threatens the credibility of any claims of reliability or validity of the questionnaire or the model upon which it is based.

Drehmer and Dekleva (1992) presented an alternative empirically-determined maturity interpretation of the SEI key practice items. That study was limited to items defining the repeatable (level 2) and defined (level 3) SEI maturity classifications. The present study extends the scope of Drehmer and Dekleva to determine whether the key questionnaire items form a cumulative hierarchy necessary to establish a pattern of growth and maturity. The relative validity of the Humphrey, Kitson and Kasse and the Drehmer and Dekleva maturity frameworks are also examined. Item response theory, and particularly the Rasch model, provides an ideal mechanism for representing the structure of scale data (Rasch 1960/1980; Wright 1977; Wright and Masters 1982; Wright and Stone 1979).

2. METHOD

2.1 Analysis

The process of any measurement involves the comparison of a stimulus to a set of standards. In this case, the stimulus is the respondent’s software engineering practice and the standards are questionnaire items. Traditionally, a simple count of questionnaire items endorsed has been assumed to provide a measure of the strength of the phenomenon being assessed.

It turns out, however, that the count of endorsed items is a sufficient indicator only under some fairly restrictive conditions. To convert this count to a measure, it matters which items have been endorsed. We would expect that items representing maturity stages already passed by subjects would be endorsed while items representing maturity levels not yet reached would be rejected at least in a probabilistic sense. This modern approach to measurement owes its logic and underlying mathematics to Georg Rasch. The Rasch model is not a data-analytic model based on data covariance structures, but rather a definition of the necessary conditions for measurement.

The fundamental idea behind our application of the Rasch model is that each SEI key questionnaire item represents a particular level of software engineering maturity, $\delta_i$. When an organization endorses an item, that organization is stating that the maturity of its own software engineering practices exceeds the maturity level described by the item. If we conceptualized a continuum from low maturity to high maturity, it would be possible to locate both items, $\delta_i$, and organizations, $\beta_n$, on the same scale. This is analogous to using a ruler for making physical measurements: an object is compared to calibrated markings ($\delta_i$) of length on a stick to determine its length ($\beta_n$).
Table 1. SEI Software Process Maturity Model

<table>
<thead>
<tr>
<th>Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>The organization lacks sound management practices. Schedules, budgets, functionality, and system quality are unpredictable. The success of projects depends on heroic commitments of individuals rather than organizational capability.</td>
</tr>
<tr>
<td>Level 1</td>
<td>Project standards and management techniques are implemented. Project planning and management is based on past experience enabling organizations to repeat successful practices used on previous projects. The Repeatable maturity level provides a stable, managed, and controllable environment with realistic project plans. Project managers track costs, schedules, and functionality. They are able to identify problems in meeting plans and commitments.</td>
</tr>
<tr>
<td>Repeatable</td>
<td>Defined level is achieved by the organizations with a standard and documented process for both software engineering and management. A group for permanent evaluation and improvement of software engineering and management process is organized, training program for staff and managers is implemented, and software quality is tracked. Both projects and processes are stable; costs, schedules, functionality, and quality are under control.</td>
</tr>
<tr>
<td>Level 2</td>
<td>The organizations at the Managed maturity level collect quantitative productivity and quality measurements in an organization-wide process database. The well-defined and consistent metrics enable the evaluation of processes and products with the goal to narrow the variation and manage the effect of learning curve of a new application domain. The ability of operating within measurable limits enables level four organizations to predict process and product quality within the quantitative bounds.</td>
</tr>
<tr>
<td>Managed</td>
<td>Organizations at the Optimizing level are focused on process improvement. Statistical evidence is used to advance the existing process and to identify best new methods and technologies with the goal of preventing the defects. Defects are studied and processes strengthened to avoid recurrence of known types of defects and disseminate lessons learned throughout the organization.</td>
</tr>
<tr>
<td>Level 3</td>
<td>Level 4</td>
</tr>
<tr>
<td>Level 4</td>
<td>Level 5</td>
</tr>
</tbody>
</table>

Figure 1. Response Pattern Zones
If the items were ordered from low maturity to high maturity, one would expect that an organization would be located in a transition zone between places where all items had been endorsed and where no items had been endorsed.

The difference between the item's maturity level, $\delta_i$, and the organization's maturity, $\beta_v$, should describe the odds that the $i$th item will be endorsed by the $v$th organization.

Let $P(X_{iv}=1|\beta_v,\delta_i)$ represent the probability of endorsing the $i$th item by the $v$th organization. Then $1 - P(X_{iv}=1|\beta_v,\delta_i)$ would represent the probability of not endorsing the item. One often used formulation for expressing such probability data is the odds ratio, the ratio of endorsements to non-endorsements, i.e., $P / (1 - P)$ [subscripts omitted]. The Rasch model specifies that this odds ratio should govern the differences between the item's calibration and the organization's measure. To create an equal interval and linear scale in item calibration units, the natural logarithm of the odds, a logit, was set equal to the difference between an item's difficulty, $\delta_i$, and a respondent's ability, $\beta_v$, as expressed in equation 1.

$$\ln\left(\frac{P_{iv}}{1-P_{iv}}\right) = \beta_v - \delta_i,$$

(1)

The expected probability of endorsing an item may be derived from equation 1 and is a simple function of ability and item difficulty as shown in equation 2. This is the common expression for the Rasch model.

$$P(X_{iv}=1|\beta_v,\delta_i) = \frac{\exp(\beta_v - \delta_i)}{1 + \exp(\beta_v - \delta_i)}$$

(2)

An organization's location on the variable should indicate how far along that variable particular scaled items should be endorsed. Similarly, an item's location on the variable should determine in a probabilistic sense which organizations will endorse it. When an assessment of item and organization fit indicates congruence with this model, strong inferences about construct validity and measurement precision are possible.

One feature that distinguishes the Rasch model from other item response theory (IRT) models is the separability of item and person parameters. Rasch (1960/1980) and others have shown that the Rasch model is the only logistic model that leads to algebraically independent estimates for person and item parameters. This leads to very desirable features in that it is the only method that provides item calibrations that are invariant across persons and person measurements that are invariant across items (Rasch, 1960/1980; Wright and Masters 1982; Wright and Stone 1979). McRae quipped,

"The appropriateness of this specification is perhaps most obvious when considering the failure to achieve it in physical measurement: The yardstick which expands with increases in temperature is not a useful tool for comparing the heights of Athabaskan and Navajo Native Americans. One can easily imagine the fanciful theories that users of this yardstick would develop to account for the greater height of the Navajo (1991, p. 423)."

Rasch's separability theorem also leads to sufficient statistics for both person and item parameter estimates. This means that the estimates extract all of the relevant information from the data. Any residual differences between the observed data and that predicted by the model are known to be independent and can therefore be used to test the validity of the model.

2.2 Subjects

Two cohorts of practicing software maintainers volunteered to participate in this study. The first cohort consisted of forty-four surviving respondents (from an original base of sixty-two who were members of a third-round Delphi panel in a larger study conducted by Dekleva (1992). The second cohort of subjects consisted of thirty-nine respondents who completed a mail survey designed specifically for this investigation.

Each subject was an experienced software maintainer with an average of fifteen years in MIS of which eleven years had been spent in software maintenance or management of maintenance and therefore was qualified as an appropriate subject. The first cohort were solicited from persons who attended the 1990 and 1991 Software Management Association professional conferences and the second cohort were attendees of the 1992 meeting of the same conference. Each indicated a high level of maintenance problem awareness and interest in maintenance in general. With the exception of one participant from Australia and one from Great Britain in the first cohort, all were from the United States or Canada.
2.3 Instruments

The instrument consisted of thirty-eight items described as "the key questions" to determine the software engineering maturity levels "repeatability, level 2" (items 1-12), "defined, level 3" (items 13-26) and "managed, level 4" (items 27-38) (Humphrey, Kitson and Kasse 1989). The first twenty-six items were presented to the first cohort as part of a third round Delphi survey as a separate section with the following heading: "The following are supplementary questions which should enable us to provide important additional feedback. The questions investigate your MIS department in general, not just the maintenance function."

The second cohort completed a mailed survey consisting of sixty-one items, thirty-eight of which were key items representing the SEI repeatable, defined and managed maturity thresholds. Since no organization was believed to pass the optimizing threshold, none of these items were used. It was the belief that the practices defined by these items would rarely occur and would not be revealing of the underlying structure of process maturity. The full text of the thirty-eight survey items is presented in the appendix to this paper.

Each item from the questionnaire was coded "1" if it was endorsed by the respondent and coded "0" if it was not endorsed. Rasch item calibrations were obtained from the BIGSTEPS, Version 2.31 (Wright and Linacre 1992).

3. RESULTS

A preliminary analysis was undertaken to determine whether the two cohorts were sufficiently similar to warrant pooling them to obtain more stable estimates of item locations than could be done with each sample separately. The twenty-five items from the SEI Software Process Maturity Scale (Humphrey, Kitson and Kasse 1989) used by Drehmer and Dekleva were calibrated using BIGSTEPS, Version 2.31 (Wright and Linacre 1992) for each cohort separately. The locations (MEASURE) under each analysis for corresponding items as shown in Figure 2 were significantly correlated with each other (r(25) = .847, p < .001).

The two cohorts were then pooled to form one sample and the twenty-five items were recalibrated. The correlations of item locations for each cohort and the combined sample were r(25)=.966 (p < .001) and r(25) = .952 (p < .001), respectively. The reliability estimate of item calibrations was .95 based on the pooled sample.

The primary analysis consisted of a Rasch calibration of all items and all subjects. Five items were deleted due to a lack of fit; two subjects appeared to provide response patterns considerably different from others in the sample and each was eliminated in order to estimate an uncontaminated location for each remaining item.

Table 2 provides the most stable and best Rasch estimate of the item's (software engineering practice) location (MEASURE) on the underlying maturity variable using the available data. The measure of location is the primary information of interest in this analysis. Every measure is an estimate and the uncertainty or precision of that estimate is described by its standard error (ERROR). An additional set of fit statistics (INFIT and OUTFIT) describes how well the data fit the model. These statistics answer the question about whether the item belongs in this construct.

If one conceptualizes a variable, a particular respondent would be expected to have a zone of endorsement at the end of the scale, a zone of rejection at the other end, and a zone of transition in the middle. Two fit statistics are provided and each assesses a different character of possible misfit. Both of these statistics are presented in their standardized form. OUTFIT is an approximately normally-distributed index based on a sum of squared residuals and is particularly sensitive to outlying values (responses located away from the center). Its expected value is zero. Negative OUTFITs indicate a response pattern that is more stable and generally a better-fitting pattern than would be expected while positive OUTFITs indicate a more random pattern of responses than would be expected from the model in the endorsement and rejection zones. INFIT provides an alternative fit assessment that is more sensitive to inlying observations and is relatively insensitive to the effects of outliers. INFIT is particularly sensitive to the pattern of responses in the zone of transition. INFIT is also approximately normally distributed with an expectation of zero and a standard deviation of one.

INFITs or OUTFITs greater than two would not be expected more than 5% of the time if the data conforms to the model. Consequently, misfitting items were eliminated from further analysis.

Several items were eliminated from the final scale. Item 13, "There is a software engineering process group function," was discarded from our final version of the scale due to a lack of fit. It is believed that the phrase "software engineering process group" was not properly understood by respondents if they had not heard of the SEI work before. Item 9, "Senior management has a mechanism for the
status review of software development projects'' did not fit the model, although it was very frequently endorsed. Since it seemed to add very little to understanding the model, and because of its lack of fit, it too was eliminated. Two items dealing with statistical reporting, ``Statistics on software design errors are gathered'' and ``Statistics on software code and test errors are gathered'' were also found to not represent the underlying construct represented by the rest of the items. These were also eliminated as was ``The Software Quality Assurance (SQA) function has a separate management reporting channel.'' The reliability of the final scale of thirty-three items was .93.

4. DISCUSSION

The SEI definition of software process maturity levels as described by Humphrey, Kitson and Kasse seemed arbitrary and may be improved. While SEI reported no psychometric technical data for the performance of their scale, many items were misplaced in terms of defining a strict deterministic hierarchy. The items numbered 1 through 12 were used by SEI to demarcate the transition from level 1 to level 2, items numbered 13 through 26 differentiate level 2 from level 3, while the remainder of the items were used to distinguish companies transitioning from level 3 to level 4.
SEI requires that 90% of the items be answered affirmatively to progress to the next stage of maturity.

Table 2 displays the items in the calibration order produced by the present analysis. If one examines an item's calibration within the transition levels to which it belongs, a different picture emerges. The box and whisker display in Figure 3 presents the distribution of item calibrations for items in each SEI level. The box marks the interquartile range of the items and the whiskers denote the range of the tails of the distributions. An examination of these diagrams reveals generally increasing calibrations along the maturity variable for each succeeding level. This would support a maturity model. However, the heterogeneous item clustering within a maturity level and the large overlap among the levels indicates that this particular notion of maturity is substantially unsupported and is very vulnerable.

If one examines the continuity of content of items along the scale described by the Rasch analysis, one might speculate that the clustering of items represents a continuous progression rather than three distinct levels. One can entertain the notion that a meaningful growth hierarchy is represented by these items, particularly since the Rasch model requires

---

### Table 2. Calibrations, Standard Errors and Fit Statistics of Maturity Scale

<table>
<thead>
<tr>
<th>ITEM &amp; LEVEL*</th>
<th>MEASURE</th>
<th>ERROR</th>
<th>INFIT</th>
<th>OUTFIT</th>
</tr>
</thead>
<tbody>
<tr>
<td>38</td>
<td>9.18</td>
<td>1.45</td>
<td>Max.</td>
<td>estim.</td>
</tr>
<tr>
<td>27</td>
<td>7.64</td>
<td>.77</td>
<td>-.2</td>
<td>-.1</td>
</tr>
<tr>
<td>32</td>
<td>7.64</td>
<td>.77</td>
<td>-.2</td>
<td>-.2</td>
</tr>
<tr>
<td>15</td>
<td>7.54</td>
<td>.45</td>
<td>.5</td>
<td>1.2</td>
</tr>
<tr>
<td>25</td>
<td>7.08</td>
<td>.41</td>
<td>-.5</td>
<td>-.7</td>
</tr>
<tr>
<td>14</td>
<td>6.94</td>
<td>.37</td>
<td>1.2</td>
<td>1.1</td>
</tr>
<tr>
<td>34</td>
<td>6.75</td>
<td>.58</td>
<td>-.6</td>
<td>-.4</td>
</tr>
<tr>
<td>7</td>
<td>6.63</td>
<td>.35</td>
<td>.9</td>
<td>-.1</td>
</tr>
<tr>
<td>26</td>
<td>6.49</td>
<td>.34</td>
<td>-.1</td>
<td>-.3</td>
</tr>
<tr>
<td>20</td>
<td>6.44</td>
<td>.54</td>
<td>-.3</td>
<td>-.2</td>
</tr>
<tr>
<td>29</td>
<td>6.17</td>
<td>.50</td>
<td>-.4</td>
<td>-.2</td>
</tr>
<tr>
<td>36</td>
<td>6.12</td>
<td>.50</td>
<td>-.9</td>
<td>-.6</td>
</tr>
<tr>
<td>21</td>
<td>5.48</td>
<td>.29</td>
<td>1.3</td>
<td>.3</td>
</tr>
<tr>
<td>33</td>
<td>5.33</td>
<td>.42</td>
<td>-1.2</td>
<td>-.2</td>
</tr>
<tr>
<td>35</td>
<td>5.00</td>
<td>.40</td>
<td>.9</td>
<td>.4</td>
</tr>
<tr>
<td>4</td>
<td>4.95</td>
<td>.27</td>
<td>-.6</td>
<td>-.8</td>
</tr>
<tr>
<td>34</td>
<td>4.84</td>
<td>.40</td>
<td>1.4</td>
<td>1.8</td>
</tr>
<tr>
<td>2</td>
<td>4.63</td>
<td>.27</td>
<td>.1</td>
<td>.2</td>
</tr>
<tr>
<td>19</td>
<td>4.38</td>
<td>.27</td>
<td>.2</td>
<td>.0</td>
</tr>
<tr>
<td>20</td>
<td>4.37</td>
<td>.27</td>
<td>-.3</td>
<td>-.2</td>
</tr>
<tr>
<td>31</td>
<td>4.36</td>
<td>.39</td>
<td>-.7</td>
<td>-1.0</td>
</tr>
<tr>
<td>6</td>
<td>4.29</td>
<td>.27</td>
<td>-.6</td>
<td>-.4</td>
</tr>
<tr>
<td>28</td>
<td>4.23</td>
<td>.38</td>
<td>.7</td>
<td>.0</td>
</tr>
<tr>
<td>5</td>
<td>4.21</td>
<td>.27</td>
<td>1.0</td>
<td>.3</td>
</tr>
<tr>
<td>17</td>
<td>4.06</td>
<td>.27</td>
<td>1.0</td>
<td>1.7</td>
</tr>
<tr>
<td>3</td>
<td>3.76</td>
<td>.28</td>
<td>.6</td>
<td>.0</td>
</tr>
<tr>
<td>10</td>
<td>3.51</td>
<td>.28</td>
<td>-.1</td>
<td>1.7</td>
</tr>
<tr>
<td>16</td>
<td>3.45</td>
<td>.28</td>
<td>-2.3</td>
<td>-1.7</td>
</tr>
<tr>
<td>23</td>
<td>3.40</td>
<td>.29</td>
<td>.4</td>
<td>-.4</td>
</tr>
<tr>
<td>24</td>
<td>2.93</td>
<td>.31</td>
<td>-2.1</td>
<td>-1.8</td>
</tr>
<tr>
<td>11</td>
<td>2.84</td>
<td>.31</td>
<td>.0</td>
<td>-.6</td>
</tr>
<tr>
<td>22</td>
<td>2.64</td>
<td>.32</td>
<td>.7</td>
<td>.0</td>
</tr>
<tr>
<td>12</td>
<td>1.93</td>
<td>.37</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Items 1-12 were SEI Level 2 items; items 13-16 were Level 3, and items 27-38 were Level 4.
that, in order to endorse a higher item, there must be a high probability that lower level items were also endorsed. A joint distribution of items and respondents is given in Figure 4.

4.1 Stage A: Reviews and Change Control

An examination of the item content along the calibrated maturity scale on the ordinate axis should reveal how practices change as the software engineering matures. Consistent with Drehmer and Dekleva, the lowest items describe practices of performing design and code reviews and the use of mechanisms for controlling code, requirements, and design changes. The establishment of these controls lifts an environment to the first stage of software process maturity and occupies measures from zero through about 3.4 logits (log odds units).

4.2 Stage B: Standard Process and Project Management

It appears that the next level focuses on standardization of software development process and implementation of project management practices. The two efforts seem to be
<table>
<thead>
<tr>
<th>Organization</th>
<th>Design errors projected</th>
<th>Training review leaders</th>
<th>Code, test errors projected</th>
<th>Process metrics database</th>
</tr>
</thead>
<tbody>
<tr>
<td>X X X X X X</td>
<td>SQA sample verification</td>
<td>Developers training required</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Forecast remaining errors</td>
<td>Profiles of software size</td>
<td>Adequacy of regression test</td>
<td>Review efficiency analyzed</td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Design and code coverage</td>
<td>Test coverage measured</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Compliance with standards</td>
<td>Error cause analysis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Size estimated</td>
<td>Code review standards</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Configuration control</td>
<td>Design review items tracked</td>
<td>Design review data analyzed</td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Software cost estimated</td>
<td>New technology intro. managed</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Software development scheduled</td>
<td>Standards documented, used</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Formal management review</td>
<td>Managers sign off</td>
<td>Development standardized</td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Design changes controlled</td>
<td>Requirements change control</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Code reviews conducted</td>
<td>Design reviews conducted</td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Code changes controlled</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>X X X X X X</td>
<td>Organization</td>
<td>Key software practices</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 4. Maturity Map of Organizations and Key Software Practices**

Simultaneous, which contradicts the SEI model. Paulik, Curtis, and Chrissis (1991, p. 24) support Humphrey’s (1988) position in saying “it may seem easier to define and implement an engineering process than a management process (especially in the eyes of technical people), but without management discipline, the engineering process is sacrificed to schedule and cost pressures.”

Soon after the managers start signing off and reviewing before making contractual commitments, formal scheduling and estimating procedures are implemented. Similarly, soon after a standard software process is realized, a need for a mechanism used for managing and supporting the introduction of new technologies is recognized. This stage runs from about 3.4 to 4.3 logits.

### 4.3 Stage C: Review Management and Configuration Control

The practices then smoothly shift to focus on managed outcomes of the reviews and information produced when standards were not fully met. Review data are analyzed, action items from code and design reviews are tracked to
closure, and code review standards are applied. In addition, software configuration control becomes a function practiced in every project and a formal procedure for software size estimation is introduced. Stage C spans from 4.3 to just under 5 logits.

4.4 Stage D: Software Process Improvement

This represents the endpoint of the middle maturity stages. These stages generally deal with the analysis and improvement of software process and include stages B, C and D. Perhaps the evolutionary push was to develop a process that produced fewer action items. The process is systematically being improved, analysis of errors is extended to identify related process inadequacies, and a mechanism is put in place to ensure compliance with the software process standards. It appears that a standard software process is implemented at Stage B and that this process is systematically analyzed and perfected at maturity Stage D, which runs from 5 to 5.5 logits on this empirically derived scale.

4.5 Stage E: Management of Review and Test Coverages

A noticeable gap between Stages D and E separates the middle from the higher maturity stages where advanced engineering and management practices are gradually implemented. At Stage E, the coverages of design, code, and test reviews are measured and recorded. In other words, reviews are not only practiced but their coverage becomes systematically observed. This indicates that the significance of reviews becomes apparent at this point of maturation and data accumulated at this level enable further progression to Stage F. This stage occupies locations from about 6 logits to 6.2 logits on the maturity scale.

4.6 Stage F: Analysis of Measurements

Data accumulated at the lower level now enable the introduction of review efficiency analysis for every project, while an errors database enables assessments of remaining errors distribution. A mechanism is also implemented to assure the adequacy of regression testing. The scope of measurements is further expanded. For example, maintenance of software size profiles for each configuration is initiated. The base of this stage is located at about 6.4 logits and extends to about 6.8 logits.

4.7 Stage G: Advanced Practices

Some of the practices at this level are above the maturity level of the most mature organizations in our sample. Six practices describing this level represent two issues. It comes as a surprise that an organization needs to progress in its software process maturity all the way up to this level to realize that training should be required. Perhaps the analysis at the preceding level helps managers recognize this need. The other issue is further extension in quality reliability assurance practices. For example, the sampling adequacy for quality assurance is insured and code, test, and design errors are projected and analyzed. Finally, a process measurements database is established at this level and process measurements are gathered for all projects. This activity seems to prepare an organization to climb to the process optimization maturity level and possibly other higher levels not investigated in this study. Anything above 6.9 logits is considered to belong to Stage G.

David Andrich (1988, p. 10) has pointed out that "devising a measuring instrument is as important in what it teaches about the variable as are the subsequent acts of measurement using the instrument." For example, management may have been asked to sign off on a project (item 10) without an evaluation against schedules and estimations (items 4, 5 and 6). Similarly, scheduling appears at a lower level of maturity than estimation. At first thought this seems unreasonable in that management is put in the impossible position of taking responsibility for a project without any means of evaluating the consequences of its actions. However, it might point out the evolutionary need to develop the tools for such an evaluation showing a higher level of maturity. It might also show that a fundamental change in what tasks are done has not changed, but how that task is accomplished may change considerably throughout the maturity cycle.

The "evolutionary necessity" hypothesis can also be supported by noticing that once standards and controls are established, one still needs to insure compliance with those standards. This is indicative of yet higher levels of software engineering maturity.

5. CONCLUSIONS

In summary, this research has shown that the Software Process Maturity Scale developed by Humphrey, Kitson and Kasse does not perform in the cumulative discrete way that those researchers believed that it would. However, by fitting item responses to the Rasch model, a more precise picture of the software engineering process has been identified. By using this new definition, a meaningful measure for software engineering maturity may be made that not only makes conceptual sense, but also is supported by empirical data.

By using a simple measure derived from this analysis, and the Rasch model, a user should be able to assess the likeli-
hood that particular techniques are in use. The present authors believe that with this new definition it is not only possible to describe software engineering maturity, but it may also be possible to provide remediation for those who have not developed along its path or for those seeking to move up the ladder of maturity.

Future research should focus on an independent validation of the structure of the maturity construct as it has been redefined. This could be accomplished by taking additional items and hypothesizing where they would load. A calibration of these new items and fixing the position of the already known items will allow for appropriate tests of content hypotheses within the Rasch model framework. Attention should also be devoted to determining whether the nature of the lower level activities changes as maturity progresses.

The respondents in this study come from environments ranging from scientific programming to the development of business systems to operating systems. While these encompass a very large proportion of computer systems development tasks, still others with unique methodologies exist. Questions remain unanswered as to whether this model is applicable to projects using rapid prototyping, projects that are predominately artificial intelligence driven, such as expert systems or robotics, or those that come from very small-project working environments.

6. REFERENCES


APPENDIX

Item Text for Software Process Maturity Scale in Calibration Order.

38. Are design errors projected and compared to actual?
37. Has a managed and controlled process database been established for process metrics data across all projects?
36. Are code and test errors projected and compared to actual?
35. Is a formal training program required for design and code review leaders?
34. Is a mechanism used for verifying that the samples examined by Software Quality Assurance are truly representative of the work performed?
33. Is there a required software engineering training program for software developers?
32. Is the error data from code reviews and tests analyzed to determine the likely distribution and characteristics of the errors remaining in the product?
31. Are profiles of software size maintained for each software configuration item, over time?
30. Is there a mechanism for assuring the adequacy of regression testing?
29. Is a mechanism used for assuring the adequacy of regression testing?
28. Is a mechanism used for ensuring compliance with the software engineering standards?
27. Are design and code review coverages measured and recorded?
26. Are design and code review covers measured and recorded?
25. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
24. Is a mechanism used for periodical assessments of the software engineering process and implementing indicated improvements?
23. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
22. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
21. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
20. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
19. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
18. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
17. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
16. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
15. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
14. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
13. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
12. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
11. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?
10. Is a mechanism used for periodically assessing the software engineering process and implementing indicated improvements?

XXX Does senior management have a mechanism for the regular review of the status of software development projects?
XXX Is there a software engineering process group function?
XXX Are statistics on software design errors gathered?
XXX Are statistics on software code and test errors gathered?
XXX Does the Software Quality Assurance (SQA) function have a management reporting channel separate from the software development project management?