TECTRAMO – A Parametric Model for Estimating Information Technology Transition Rates

Barry Boehm, USC
SEI Software Engineering Symposium 2001
October 16, 2001

Outline

- Motivation
- Scope, Objectives, Definitions
- Overall Approach
- Initial Model Formulation
  - Structure
  - Elements
  - Examples of Details
- Status and Plans
Motivation

- Efficient technology transition is critical to competitive advantage
  - Michael Porter books (The Competitive Advantage of Nations)
  - Geoffrey Moore books (Crossing the Chasm)
  - Cost of delay (MIT, McNutt)
- Weak level of quantitative understanding
  - A few starting points (Redwine-Riddle, SEI)
- Rapid pace of IT change
  - Technology transition a necessity, not an option

Objectives and Scope

- An initial framework vs. a calibrated model
  - Sparse data; weak understanding
  - Prototype and iteration
- A context for collaborative efforts
  - OSD/SIS, TACOM, SEI, USC/CeBASE, others
- Start in IT field
  - Strong need w.r.t. competitive advantage, quantitative understanding, pace of change
Definitions

- "Technology:" hardware, software, best practices
  - Architectures, components, methods, tools
- "Transition:" progress along milestones from idea emergence to general practice
  - Technology Readiness Levels
  - Redwine-Riddle Maturity Points
- Transition Rate Drivers: defined below

General Technology Readiness Levels

<table>
<thead>
<tr>
<th>Technology Readiness Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Basic principles observed and reported.</td>
<td>Lowest level of technology readiness. Scientific research begins to be translated into applied research and development. Examples might include basic studies of a technology's basic properties.</td>
</tr>
<tr>
<td>2. Technology concept and/or application formulated.</td>
<td>Concept begins. Once basic principles are observed, practical applications can be derived. The application is speculative and there is no proof or detailed analysis to support the description. Examples are still limited to paper models.</td>
</tr>
<tr>
<td>3. Analytical and experimental critical function and/or characteristic proof of concept.</td>
<td>Active research and development is initiated. This includes analytical studies and laboratory studies to physically validate analytical predictions of separate elements of the technology. Examples include components that are not yet integrated or representative.</td>
</tr>
<tr>
<td>4. Component and/or broadboard validation in laboratory environment.</td>
<td>Basic functional components are integrated to establish that the pieces will work together. This is relatively &quot;low fidelity&quot; compared to the eventual system. Examples include integration of &quot;lab box&quot; hardware in a laboratory.</td>
</tr>
<tr>
<td>5. Component and/or broadboard validation in a relevant environment.</td>
<td>Fidelity of broadboard technology increases significantly. The basic technological components are integrated with reasonable realistic supporting elements so that the technology can be tested in a simulated environment. Examples include &quot;high fidelity&quot; laboratory integration of components.</td>
</tr>
<tr>
<td>6. System/subsystem model or prototype demonstration in an operational environment.</td>
<td>Representative model or prototype system, which is well beyond the broadboard tested for TRL 5, is tested in a relevant environment. Represents a major step up in a technology's demonstrated readiness. Examples include testing a prototype in a high fidelity laboratory environment or in an actual operational environment.</td>
</tr>
<tr>
<td>7. System prototype demonstration in an operational environment.</td>
<td>Prototype near or at planned operational system. Represents a major step up from TRL 6. Includes demonstration of an actual system prototype in an operational environment, such as in an aircraft, vehicle, or space. Examples include testing the prototype in a test bed several miles away.</td>
</tr>
<tr>
<td>8. Actual system tested and &quot;flight qualified&quot; through test and demonstration.</td>
<td>Technology has been proven to work in its final form and under expected conditions. In almost all cases this TRL represents the end of true system development. Examples include developmental test and evaluation of the system and in its intended weapon system to determine if it meets design specifications.</td>
</tr>
<tr>
<td>9. Actual system flight proven through successful mission operations.</td>
<td>Actual application of the technology in its final form and under mission conditions, such as those encountered in operation and evaluation. In almost all cases, this is the end of the &quot;flight proving&quot; aspects of true system development. Examples include using the system under operational mission conditions.</td>
</tr>
</tbody>
</table>
# Software Readiness Levels

<table>
<thead>
<tr>
<th>Technology Readiness Level</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Functionality conceptual.</td>
<td>Lowest level of software readiness. Basic research begins to be translated into applied research and development. Examples might include a concept that can be implemented in software or analyze studies of an algorithm's basic properties.</td>
</tr>
<tr>
<td>2. Technology concept and/or preliminary prototype.</td>
<td>prototype is studied and there is no prior or detailed analysis to support the assumptions. Examples are still limited to studies.</td>
</tr>
<tr>
<td>3. Analytical and experimental critical function and characteristic proof of concept.</td>
<td>Active research and development is initiated. This includes analytical studies to produce code that validates analytical predictions of separate software elements. Examples include software prototypes that are not yet integrated or representative but satisfy an operational need. Algorithms run on a single processor in a laboratory environment.</td>
</tr>
<tr>
<td>4. Functionality demonstrated in a laboratory environment.</td>
<td>Basic software components are integrated to establish that they work together. They are relatively primitive with regard to efficiency and may be operated in the unusual system. System software architecture development initiatives to include interoperability, maintainability, scalability, and security issues. Software integrated with solvable weaknesses.</td>
</tr>
<tr>
<td>5. Functionality and performance demonstrated in a relevant environment.</td>
<td>Reliability of software assembly increases significantly. The basic software components are integrated with reasonably realistic supporting elements as that can be tested in a simulated environment. Examples include &quot;high-fidelity&quot; laboratory integration of software components. System software architecture established. Algorithm run on processing with characteristics. Expected software performance are &quot;virtual&quot; versions and configuration control tested. Verification, Validation, and Accreditation (VVA) initiatives.</td>
</tr>
<tr>
<td>6. Functionality and performance demonstrated in a realistic simulated (synthetic) operational environment.</td>
<td>Representative-model or prototype system, which is well beyond that ofTRL 5, is tested in a realistic environment. Represents a major step-up in software demonstrability/reliability. Examples include testing prototypes in a simulated environment or in a limited operational environment. Software architecture established. Algorithm run on processing with characteristics. Expected software performance are &quot;real&quot; versions and configuration control tested. Verification, Validation, and Accreditation (VVA) initiatives.</td>
</tr>
<tr>
<td>7. Functionality and performance demonstration in an operational test environment.</td>
<td>Represents a major step up from TRL 6, requiring the demonstration of an actual system prototype in an operational environment, such command post or high-end collaborative. Algorithms run on processor of the operational environment integrated with actual external entities. Software releases are &quot;real&quot; version and configuration control. Software support structures in place. Software releases are in district versions. Frequency and severity of software deficiencies require not significantly degrade functionality or performance. VVA complete.</td>
</tr>
<tr>
<td>8. Functionality, performance, and quality attributes validated in an operational environment.</td>
<td>Software has been demonstrated to work in its final form and under expected conditions. In most cases, this TRL represents the end of system development. Examples include test and evaluation of the software to its mission system or an enhanced it meets design specifications. Software releases are production version and configuration control. Software evaluations are used through support materials. Software architecture are usually integrated through support materials.</td>
</tr>
<tr>
<td>9. Functionality, performance, and quality attributes proven in an operational environment with successful accomplishment of mission operations.</td>
<td>Alpha application of the software in its final form and under mission conditions, such as those encountered in operational test and evaluation. In almost all cases, this is the end of the final &quot;buying&quot; aspects of system development. Examples include using the system under operational condition criteria. Software releases are production version as configuration control. Frequency and severity of software deficiencies are at a minimum.</td>
</tr>
</tbody>
</table>

* Quality attributes include reliability, maintainability, interoperability, scalability, and security.

From Mike Saboe, USA-TACOM

10/16/01 ©USC-CSE

---

# Redwine-Riddle and Software Readiness Levels

<table>
<thead>
<tr>
<th>Redwine-Riddle Maturation Points</th>
<th>Counterpart Software Readiness Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>0. Concept Emergence</td>
<td>1. Concept emerges</td>
</tr>
<tr>
<td>1. Clear Approach</td>
<td>2. Invention begins</td>
</tr>
<tr>
<td>Demonstrated, Analyzed</td>
<td>3. Early analysis validation</td>
</tr>
<tr>
<td>2. Usable Capabilities</td>
<td>4. Solution components integrated</td>
</tr>
<tr>
<td>Available (alpha test)</td>
<td>5. &quot;Alpha&quot; software releases</td>
</tr>
<tr>
<td>Group (beta)</td>
<td>7. Demonstrated in operational environment</td>
</tr>
<tr>
<td>4. Substantial Evidence of</td>
<td>8. Validated production version</td>
</tr>
<tr>
<td>Value</td>
<td>9. Applied under mission conditions</td>
</tr>
<tr>
<td>4a. Propagation Through</td>
<td></td>
</tr>
<tr>
<td>40% of Community</td>
<td></td>
</tr>
</tbody>
</table>

10/16/01 ©USC-CSE
Overall Approach

- Use USC 7-step model development approach
- Develop initial framework (here)
- Exploratory studies of framework, rate drivers
  - Surveys and analyses
  - Initial quantification
    - Delphi: limited data
    - SEI coordination
- Assessment of next-step options
  - Model reformulation
  - Data collection and analysis
  - Trial use

USC 7-Step Model Development Approach
- concurrency and feedback implied

- Step 1: Define project objectives and scope
- Step 2: Perform external analyses
- Step 3: Identify input factors
- Step 4: Formulate models
- Step 5: Quantify relationships
- Step 6: Validate model
- Step 7: Evaluate model

GOCOMOL book, p. 142
Outline

- Motivation
- Scope, Objectives, Definitions
- Overall Approach
- Initial Model Formulation
  - Structure
  - Elements
  - Examples of Details
    - Status and Plans

Initial Model Formulation

- Start simple, familiar
  - Redwine-Riddle and Software Readiness Levels
  - COCOMO-like orthogonal rate drivers
- Exploratory validity via literature, expert surveys, case studies
- Review and experience application
  - Mini-workshops
  - Recent experiences
  - Evaluation and iteration
Initial Model Structure

\[ TTT(i,i+1) = \text{BTTT}(i,D) \prod_{\mu=1}^n \text{TRM}(j) \]

- \( TTT(i,i+1) \): Technology Transition Time from Level \( i \) to Level \( i+1 \)
  - either Redwine-Riddle or Software Readiness
  - special possibilities: CMM levels
- \( \text{BTTT}(i,D) \): Baseline Technology Transition Time from Level \( i \) to Level \( i+1 \) in Domain \( D \)
  - shorter for VLSI chips than software processes
- \( \text{TRM}(j) \): Technology Transition Rate Multipliers
  - hypothesis: independent of level, domain

Technology Transition Rate Multipliers

1. Competition-criticality
2. Impact on current operations, power bases
3. Number of concurrent innovations involved
4. Adopter change/risk-aversion
5. Payback realization speed
6. Regulatory obstacles or incentives
7. Degree of executive support
8. Factor Endowment (human, knowledge, capital, infrastructure)
9. others ...
Cross-Check: Relation to Porter Factors
- From The Competitive Advantage of Nations

P1. Factor Endowment
- Same as 8 (added in cross-check)
P2. Demand Composition (site, sophistication)
- 4. Adopter risk-aversion; 5. Payback speed
P3. Related and Supporting Industries
- 3. Concurrent innovations; 8. Factor endowment
P4. Firm Strategy, Structure, Rivalry
- 1. Competition; 4. Internal risk-aversion,
  7. Executive Support

Example Rating Scale and Multipliers:
1. Competition-Criticality

<table>
<thead>
<tr>
<th>Rating Level</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rate</td>
<td>Low Relevance</td>
<td>Modest Relevance</td>
<td>General Hygiene Factor</td>
<td>Major Competitive Factor</td>
<td>Survivability Critical</td>
</tr>
<tr>
<td>Multipliers</td>
<td>2.25</td>
<td>1.5</td>
<td>1.0</td>
<td>0.687</td>
<td>0.444</td>
</tr>
<tr>
<td>Example:</td>
<td>Years from CMM 2 to 3</td>
<td>6.75</td>
<td>4.5</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Organization</td>
<td>E-commerce startup</td>
<td>E-commerce</td>
<td>E-commerce</td>
<td>Source Selection Factor</td>
<td>Mandated CMM Level 3</td>
</tr>
</tbody>
</table>

Example:
Organization: E-commerce startup
E-commerce CMM level 3
Survivability critical.
Example Rating Scale and Multipliers:

3. Number of Concurrent Innovations

<table>
<thead>
<tr>
<th>Rating</th>
<th>Level</th>
<th>Very Low</th>
<th>Low</th>
<th>Nominal</th>
<th>High</th>
<th>Very High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rating</td>
<td>0-1 concurrent innovations</td>
<td>Simple</td>
<td>2: simple, uncoupled</td>
<td>2: modest coupling, complexity</td>
<td>3-4: modest coupling, complexity</td>
<td>3-4: high coupling, complexity</td>
</tr>
<tr>
<td>Multipliers</td>
<td>0.5</td>
<td>0.707</td>
<td>1.0</td>
<td>1.414</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>Example</td>
<td>Plug-compatible hardware</td>
<td>Simple software upgrade</td>
<td>New missile sensor, homing</td>
<td>New missile sensor, homing, controls, GPS</td>
<td>Early Ada, compilers, SDI</td>
<td></td>
</tr>
</tbody>
</table>

Status and Plans

- Refined via collaborative discussions
  - Rich Turner, OSD/SIS
  - Mike Saboe, USA/TACOM
  - Caroline Graettinger, SEI
- Framework used in formulating software technology transition strategies
  - USC-UMD Center for Empirically-Based Software Engineering (CeBASE)
  - NASA High Dependability Computing Initiative
- Looking for data, collaborators, PhD student