Lecture Notes #27: Software Risk Management

- Software risks:
  - What can go wrong?
  - What is the likelihood?
  - What will be the damage?
  - What can be done about it?

- Risk analysis and management are a set of activities that help a software team to understand and manage uncertainty about a project.

Copyright Saiedian © 2006 (For Exclusive Use of EECS SE Students)
Risk: Definition and Attributes

- Risk is the uncertainty associated with the outcome of a future event and has a number of attributes:
  - Uncertainty (probability)
  - Time (future event)
  - Potential for loss (or gain)

- Multiple perspectives, e.g.,
  - Process perspective (development process breaks)
  - Project perspective (critical objectives are missed)
  - Product perspective (loss of code integrity)
  - User perspective (loss of functionality)
A Risk vs a Problem

- A problem is a condition that exists and is undesirable. Thus it is a value judgment made upon the merits of the current condition.

- A risk suggests a possible, future undesirable condition (or consequence). Thus it is a value judgment made upon the potential implications of the current conditions.

- Risks that are not identified (or identified but not prevented) will later become problems

- Note: risks are not always problems; they can be beneficial
Risk Management

- The process by which a course of action is selected that balances the potential impact of a risk weighted by its probability of occurrence and the benefits of avoiding (or controlling) the risk

- Risk management life cycle:
  - **Identify** (risk identification)
  - **Analyze** (risk analysis)
  - **Plan** (contingency planning)
  - **Track** (risk monitoring)
  - **Control** (recovery management)

- Risk management: Reactive vs Proactive
Reactive Risk Management

- Project team reacts to risk when they occur
- Fix on failure: resources are found and applied when the risks strike
- Crisis management: failure does not respond to applied resources and the project is in jeopardy
Proactive Risk Management

- Formal risk analysis is performed
- An attempt is made to correct the root causes
  - statistical SQA
  - developing skills to manage changes
  - examining risks sources beyond the bounds of the software
Risk Projection

- Estimate the probability of occurrence
- Estimate the impact on the project on a particular scale, e.g.,
  1 ⇔ low impact (negligible)
  2 ⇔ medium impact (marginal)
  3 ⇔ high impact (critical)
  4 ⇔ very high impact (catastrophic)
- Build a risk table and sort by probability and impact
### Building A Risk Table: Example

<table>
<thead>
<tr>
<th>Risk</th>
<th>Probability</th>
<th>Impact</th>
<th>RM Plan</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size estimate may be too low</td>
<td>70%</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td>Less reuse than planned</td>
<td>60%</td>
<td>3</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Funding will be lost</td>
<td>30%</td>
<td>4</td>
<td>...</td>
</tr>
<tr>
<td>More users than planned</td>
<td>5%</td>
<td>1</td>
<td>...</td>
</tr>
</tbody>
</table>
RMMM: Risk Mitigation, Monitoring, and Management

- **Mitigation**: how can we avoid the risk?
- **Monitoring**: what factors can we track that will enable us to determine if the risk is becoming more or less likely?
- **Management**: what contingency plans do we have if the risk becomes a reality?
Some Common Root Causes of Risk

- Unforeseen events (e.g., funding terminated; project leader resigns)
- Imprecise estimates (size estimates too low)
- Erroneous assumptions (can reuse earlier artifacts)
- Unrecognized dependencies
- Technology
Risk Top Ten List (Caper Jones, 1994)

- Inaccurate metrics
- Inadequate measurement
- Excessive schedule pressure
- Management malpractice
- Inaccurate cost estimating
- Silver bullet syndrome
- Creeping user requirements
- Low productivity
- Canceled projects
Pressman’s Risk List

- **Project risks**: threaten the project plan (e.g., cost overrun)
- **Technical risks**: threaten the quality and timeliness of the product (e.g., specification ambiguity)
- **Business Risks**: threaten the viability of the product (e.g., product not aligned with business strategy or losing upper management support, or losing budget)
Risk Identification

A systematic process that prepares a list of potential threats to the software project. It uses:

- Organizational/project checklist
- Individual concerns
- Past experiences
- “What-if?” questions
- Indicators from the current situation
Documenting Risks

- Use a “risk Identification Form”
- Fields may include:
  - ID
  - Description
  - Category
  - Probability
  - Potential consequences
  - Indicator
  - Avoidance approach
  - Related risks
  - Action items
Risk Impact Assessment

- Determine the probability for each risk
- Determine the impact (e.g., critical? marginal? negligible?)
- Build the risk table
- Compute the risk exposure for each risk:
  \[ RE = P \times C \]
  where:
  - \( P \) is the probability of occurrence for a risk
  - \( C \) is the cost to the project

The total risk exposure for all risks provides a means to adjust the final costs estimates for a project and/or to predict the probable increase in (staff) resources required at various points.
Risk Impact Assessment: An Example

**Risk identification:** Only 60% of the software components planned for reuse will be integrated into the new software. The remaining functionality must be developed.

**Risk probability:** 85% (likely)

**Risk impact:** 180 reusable components were planned. If only 60% can be used, 72 components have to be developed from scratch (in addition to other components). Since the average size of a component is 80 LOC and the average cost for a LOC is $85.00, the overall cost (impact) will be: $489,600.00.

**Risk exposure:** $RE = .85 \times $489,600.00 = $416,160.00

Copyright Saiedian © 2006 (For Exclusive Use of EECS SE Students)
Lecture #28: Software Engineering

Ethics

- Credentials
- Treatment of intellectual property
- Performance of technical responsibilities
- Treatment of others
- Professional development
Common Situations You May Encounter

- Misrepresentation of abilities and experience
- Releasing programs that have not been well tested
- Using code written for another organization
- Misleading claims
- Violating contracts (or promises) with clients
- Falsifying time card or expense report related to a project
IEEE-CS/ACM SE Code of Ethics and Professional Practice

1 PUBLIC  Software engineers shall act consistently with the public interest.

2 CLIENT AND EMPLOYER  Software engineers shall act in a manner that is in the best interests of their client and employer and that is consistent with the public interest.

3 PRODUCT  Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.

4 JUDGMENT  Software engineers shall maintain integrity and independence in their professional judgment.
IEEE-CS/ACM SE Code of Ethics and Professional Practice (continued)

5 MANAGEMENT Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.

6 PROFESSION Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.

7 COLLEAGUES Software engineers shall be fair to and supportive of their colleagues.

8 SELF Software engineers shall participate in lifelong learning regarding the practice of their profession and promote an ethical approach to the practice of the profession.

Copyright Saiedian © 2006 (For Exclusive Use of EECS SE Students)