CITS5501 Software Testing and Quality Assurance
Risk

Unit coordinator: Arran Stewart

Overview

- Why look at risk?
- What is risk?
- Strategies for dealing with risk

Motivation

Why do we look at risk?

- Because every project, even the simplest, involves risks

Motivation

Why do we look at risk?

- Because every project, even the simplest, involves risks
- Because people are usually not good at allowing for them

Motivation

Why do we look at risk?

- Because every project, even the simplest, involves risks
- Because people are usually not good at allowing for them
- Because risk assessment determines what other testing and quality assurance activities you undertake

Motivation

In fact *every* technique we look at in this unit can be seen as a way to try to reduce project risks.

- Why do we have system or method specifications?
  - To try to avoid miscommunication risks – the risk that our description of something will be misunderstood because it is ambiguous, or incomplete, or contradictory
- Why do we write tests? Why do code reviews?
  - To try to discover faults *early*, and reduce the risk of expensive defect correction late in the project, or after a project is released.

Motivation

And the best answer to "When have we done enough testing?" or "When have we done enough code review?" is – when we've reduced the risk to a level that we can accept.

If some parts or aspects of a project are particularly high risk – we can dedicate extra resources or techniques to those areas to reduce the risk.

Example

- What sort of requirements documentation should we require for, say, a website?
- Is there complicated or confusing terminology involved? Is there a risk of miscommunication or misunderstanding between us and our client? (e.g. perhaps for a website to be used internally by a financial services enterprise)
  We might try to reduce the risk of miscommunication by ensuring we have *glossaries* of terms, and that client representatives and developers agree on their understanding of these.

Example

- Is the domain, or the technologies we're likely to have to use, complicated? Is there a risk we have modelled it incorrectly? (Example: software to manage telephone exchange systems; systems which are inherently concurrent, as they are notoriously difficult to reason about.)
  We might reduce risk by putting additional effort into modelling the domain and/or the system – perhaps through a notation such as UML, perhaps even through formal (i.e. mathematical) specifications of the domain and/or system.

Example

- On the other hand, if this is a website for a local kids' football club, then many of these risks are unlikely to be relevant, or to cause significant problems if they do – so effort on those activities would be misplaced.

What is risk?

- Something that ***might*** happen, which would cause **loss**
  - If it will *definitely* happen, that's not a risk – it's usually called a "constraint"
  - If it doesn't result in some kind of unwanted consequence (i.e., a loss), it's also not a risk

How can we categorize risk?

Our aim here is not to try and come up with some kind of "perfect" way of classifying risks.
The lines between our categories will almost certainly be fuzzy.

The idea is rather that these categories can be useful in **helping us think of** (or remember) **possible risks** that might occur in a project.

The hope is that if we go through categories (at least somewhat) methodically, we're more likely to identify possible risks.

What are some sorts of risk?

One categorization (Pressman):

- Project risks
  - Things that could affect the project plan or schedule
- Technical risks
  - Risks resulting from the problem being *harder to solve* than we thought
- Business risks
  - Things that threaten the viability, as a product, of the software to be built
  - (Could be commercial viability, but also applies to in-house or even open source software)

Let's see some examples.

Sorts of risk – project risks

Project risks:

- Things that could affect the *project plan and/or schedule*, causing it to slip and costs to increase
- Examples: personnel move or resign, resources turn out to be more expensive/take longer to acquire than estimated, exchange rates shift

Sorts of risk – technical risks

Technical risks:

- Risks resulting from the problem being *harder to solve* than we thought
- They threaten the quality and timeliness of the software to be produced
- If they eventuate, implementation may become difficult or impossible
- Examples: design turns out to be infeasible, dependencies have bugs, language/framework turns out to be difficult to maintain

Sorts of risk – business risks

Business risks - Things that threaten the viability, as a product, of the software to be built (even if there are no technical or project barriers to it being implemented on time and within budget) - Examples: over-estimating the market for a product; being beaten to release by a competitor

Exercise – UWA assignment/project risks

How do Pressman's categories apply to doing CSSE projects or assignment?

- Project risks (plan or schedule)
- Technical (difficulties solving problem)
- Business (viability of product)

What sort of risks can we know about?

Robert N. Charette (1989) suggests classifying risks in terms of how can (or can't) find out about them:

- Known risks
  - Those risks that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources (e.g., unrealistic delivery date)
- Predictable risks
  - Those risks that are extrapolated from past project experience (e.g., past turnover)
- Unpredictable risks
  - Those risks that can and do occur, but are extremely difficult to identify in advance

What strategies can we adopt to deal with risk?

- ***Reactive*** risk strategies
  - "Don't worry, I'll think of something"
  - The majority of software teams and managers rely on this approach
  - Nothing is done about risks until something goes wrong
    - The team then flies into action in an attempt to correct the problem rapidly ("fire fighting")
- ***Proactive*** risk strategies
  - Steps for risk management are followed
  - Primary objective is to *reduce* risk and to have a contingency plan in place to handle unavoidable risks in a controlled and effective manner

What strategies can we adopt to deal with risk?

It might seem obvious that we should be proactive.

But when we think about how the future course of a project will go, most people tend to focus on scenarios in which most things go *right*, not the scenarios where things go *wrong*.

Query – does this apply to how you plan work on projects or assignments? Or do you allow for contingencies and things going wrong?

(Some people do this more naturally than others. We will look at this in the workshop.)

Steps for risk management

1. Identify possible risks; recognize what can go wrong
2. Analyze each risk to estimate the probability that it will occur and the impact (i.e., damage) that it will do if it does occur
3. Rank the risks by probability and impact
   - Impact may be negligible, marginal, critical, and catastrophic
4. Develop a plan to manage (some of) those risks

Risk identification

- A systematic attempt to identify risks to the project
- Because if they aren't identified, how can they be planned for?
- Two main sorts of risk:
  - Generic risks
    - Risks that are common to every software project
  - Product-specific risks
    - Risks that can be identified only by those a with a clear understanding of the technology, the people, and environment specific to the software that is to be built

Risk identification (2)

- One way of identifying risks – use a *risk checklist*
- Focuses on known and predictable risks in specific subcategories

Risk checklists

Some typical categories of items on risk checklists:

- Product size – risks associated with overall size of the software to be built
- Business impact – risks associated with constraints imposed by management or the marketplace
- Customer characteristics – risks associated with sophistication of the customer and the developer's ability to communicate with the customer in a timely manner
- Process definition – risks associated with the degree to which the software process has been defined and is followed
- Development environment – risks associated with availability and quality of the tools to be used to build the project
- Technology to be built – risks associated with complexity of the system to be built and the "newness" of the technology in the system
- Staff size and experience – risks associated with overall technical and project experience of the software engineers who will do the work

A project risk questionnaire

1. Have top software and customer managers formally committed to support the project?
2. Are end-users enthusiastically committed to the project and the system/product to be built?
3. Are requirements fully understood by the software engineering team and its customers?
4. Have customers been involved fully in the definition of requirements?
5. Do end-users have realistic expectations?
6. Is the project scope stable?
7. Does the software engineering team have the right mix of skills?
8. Are project requirements stable?
9. Does the project team have experience with the technology to be implemented?
10. Is the number of people on the project team adequate to do the job?
11. Do all customer/user constituencies agree on the importance of the project and on the requirements for the system/product to be built?

Risk estimation

- Risk estimation attempts to rate, for each risk:
  - The probability that it eventuates
  - The magnitude of loss associated with the risk, should it eventuate
- The project planner, managers, and technical staff perform risk estimation so that risks can be prioritized, and limited resources be allocated where they will have the most impact

Risk tables

- A *risk table* provides a project manager with a simple technique for risk projection
- It consists of five columns
  - Risk summary – short description of the risk
  - Risk category – (see the slides on risk categorization)
  - Probability – estimation of risk occurrence based on group input
  - Categorization of impact – e.g. (1) catastrophic (2) critical (3) marginal (4) negligible
  - RMMM – pointer to a paragraph in the Risk Mitigation, Monitoring, and Management Plan

Developing a Risk Table

- List all risks in the first column (by way of the help of the risk item checklists)
- Mark the category of each risk
- Estimate the probability of each risk occurring
- Assess the impact of each risk based on an averaging of the four risk components to determine an overall impact value
- Sort the rows by probability and impact in descending order
- Draw a horizontal cutoff line in the table that indicates the risks that will be given further attention

Assessing risk impact

- Three factors affect the consequences that are likely if a risk does occur
  - Its nature – This indicates the problems that are likely if the risk occurs
  - Its scope – This combines the severity of the risk (how serious was it) with its overall distribution (how much was affected)
  - Its timing – This considers when and for how long the impact will be felt
- The overall risk exposure formula is RE $=$ P $\times$ C, where
  - P $=$ the probability of occurrence for a risk
  - C $=$ the cost to the project should the risk actually occur
- Example
  - P $=$ 80% probability that 18 of 60 software components will have to be developed
  - C $=$ Total cost of developing 18 components is \$25,000
  - RE $= .80 \times \$25{,}000 = \$20{,}000$

Risk mitigation, monitoring, and management

An effective risk strategy for dealing with risk must consider three issues

- (Note: these are not mutually exclusive)
  - Risk mitigation (i.e., avoidance)
  - Risk monitoring
  - Risk management and contingency planning
- Risk mitigation (avoidance) is the primary strategy and is achieved through a plan

Example

Example: Risk of high staff turnover

- Meet with current staff to determine causes for turnover (e.g., poor working conditions, low pay, competitive job market)
- Mitigate those causes that are under our control before the project starts
- Once the project commences, assume turnover will occur and develop techniques to ensure continuity when people leave
- Organize project teams so that information about each development activity is widely dispersed
- Define documentation standards and establish mechanisms to ensure that documents are developed in a timely manner
- Conduct peer reviews of all work (so that more than one person is "up to speed")
- Assign a backup staff member for every critical technologist