Risk Management in Software Development

Day 11 Risk management

Charoen Vongchumyen

Email : charoen.vo@kmitl.ac.th

04/2021

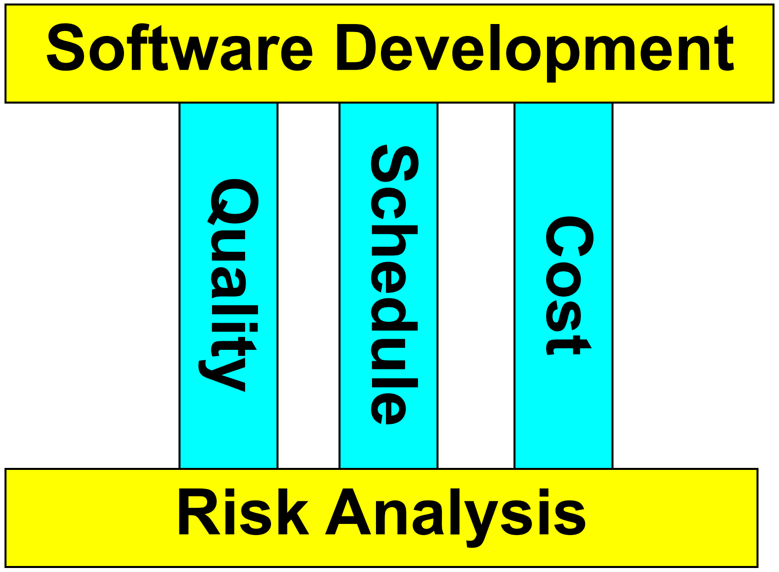
Thank to Tsuneo Yamaura

What is a Risk?

A risk in software development is anything that threatens basic of software development, namely, quality, cost, and schedule.

For example:

- **The company that will produce hardware went bankrupt.**
- **Unmatched interface with other companies.**
- **Core engineers quitted job**
- **Development office burnt to ashes.**



Definitions of Risks and Problems

- What is a risk?

Something that may happen in the future, and may result an undesirable result

- What is a problem?

A risk that is already occurred.

Why You Need to Manage Risks?

- A project without risks is not challenging, and is not worth doing
- Risk management is "insurance" of minimal money.
- Risk management clears who carries responsibility of risks.
- You will not panic if a risk becomes a problem.

Overview of Risk Management

(1) Risk Discovery

Pick up risks with discussion and brain – storming

(2) Exposure Analysis

Calculate the probability of a risk, and the expected magnitude (influence) of the risk

(3) Risk Handling

Plan how to handle risks.

(4) Risk Reduction

Define what you should do to keep risks unopened.

(5) Continuous Monitoring

Check “indexes” to monitor if a risk becomes a problem

Why You Do Not Want to Control Risks?

- A customer does not want to face risks (he is too premature to see risk.

  ⇨ That is a story of old days. Now, a customer is sensitive to risks.
- Many things are uncertain.

  ⇨ e.g., Release date is 18 to 27 months from now; 24 months in the probability of 85 %
- "Management for success" looks better than "Risk Management."

  ⇨ You will panic if a risk becomes a problem.
- You do not have enough data for risk management.

  ⇨ Many things are in common in software development.
- It is discouraging to manage risks alone

  ⇨ "Keeping a word" is more important than "making a challenging promise".

(1) Risk Discovery (part 1)

(a) Pick up risks by the following steps :

① Analyze the problem histories, and list up the worst results.

② Define “scenarios” to the worst results.

⇨ At this point, calculate the probability of each scenario.

③ Analyze the fundamental reasons that cause each scenario.

⇨ Those are the risks.

(1) Risk Discovery (part 2)

(b) Make sure that the defined risks include the following core risks (or the most common and frequent risks)

① Low – accurate estimation (or optimistic estimation)

② Unexpected expansion of functions / unfrozen functions

③ Programmers quitted job

④ Negotiation with a customer broke up.

⑤ Low productivity

(1) Risk Discovery (part 3)

(c) Define a Show - Stoppers

① A show – stopper is a huge risk that instantly erases a meaning to continue the project.

② A show – stopper is a huge risk that instantly stops the project.

⇨ You have to terminate the project if a show – stopper comes up.

You have to define what the show – stoppers are.

⇨ If you have to take care of a project that has high possibility of occurring show – stopper, you are gambling.

(2) Exposure Analysis

(a) Define a Show - Stoppers

- The “actual cost to handle the problem,” and
- The “probability of becoming a risk a problem”

(b) The total of the expected cost above is the total cost of risk handling.

⇨ If you can provide the total cost for risk handling, you will be able to overcome the risks.

⇨ It is much more important to recognize risks than to precisely calculate the probability of a risk or the total cost of risk handling.

(3) Risk Handling

Plan what you should do when a risk becomes a problem.

(a) Define "indexes" and their values that can tell if a risk becomes

(e.g., If the wind blow 20 m / sec, issue warning, and prepare for typhoon)

(b) Define the remedies to handle a risk

(4) Risk Reduction

- Reduce the probability of becoming a risk a problem.

(e.g., Volunteers patrol town to prevent kidnapping)

- Examples of Risk Reduction in software development.

  (a) Apply prototyping when designing user - interface

  (throw – away prototyping vs. reusable prototyping)

  (b) Incremental development (spiral method)

  - Sprit functions into several groups and develop each group incrementally (repeating several times).
  - You can analyze problems before they get obvious
  - You can minimize the problem in the worst case.

(5) Continuous Monitoring

- Continuously check the “indexes” to monitor if risk becomes a problem
- If environment or conditions change, always look for new “seeds” of new risks.

Risk in Software Development

Top 2 reasons that terminate software development

- **Inaccurate Estimation**
  - **Low – accurate estimation (the estimation was too short schedule, and too low cost than actual development)**
  - **Death march project (reduced to 50 % in period and cost)**
- **Unfrozen Specification**
  - **Requirement / Functional specification are not finalized.**
  - **As a system is developed (getting more visible), a customer frequently asks changes.**
  - **A customer himself does not know what he really want to do.**

Cole, Andy.1995 "Runaway Projects – Causes and Effects." Software World

Risk in estimation (part 1)

Estimation : There is no accurate methods for estimation.

- Estimation by algorithm (estimation tools)
  - Input parameters to get the estimated person – months, cost, and schedule
  - Estimated values vary 200 times.
- Estimation by LOC (Line of Code)
  - In reality, LOC estimation is much more difficult than to estimate cost and schedule

Risk in estimation (part 1)

- Estimation by FP (Function Point)
  - It is not objective. The estimation changes by person.
- Estimation base on experience
  - Not objective.

    A different person estimate based on different data.
  - The target system may not be similar to the previous system

Risk in estimation (part 2)

Why accuracy is so low ?

- Estimation timing is not right
  - Usually (Typically) when a project starts, i.e., before requirement specification is defined.
  - No specification, no precise estimation.
- Political estimation vs. Substantial estimation
  - Before estimation, the shipping date and cost have been determined. For example, game software must be shipped on Christmas sale, and the money a customer can pay has limit
- Feasibility Study
  - No feasibility study said “No, you cannot”

Risk in estimation (part 1)

Compromise in Estimate

- Apply both "estimation by algorithm," and "estimation by experience."
- As the development phase goes forward, do re – estimation.

typical software development

2 months	1.5 months	1.5 months	3 months	2 months	2 months
Requirements specification	Functional specification	Detailed design	Coding	Debugging	Testing
Quality assurance					

Risk in finalizing specification (part 1)

History of Finalizing Specification

1 st Stage

- "Changing specification is evil thing!"
  - Big fight between developers and a customer.
  - Developers insisted that "We will never ever change the specification!!"

$\downarrow$

- It is not realistic not to be able to change specification

Risk in finalizing specification (part 2)

History of Finalizing Specification

2 nd Stage

- Developers allowed specification changes but asked to pay the cost needed for changes.
  - Every time specification was changed, estimation was made.

$$\downarrow$$

- Got into endless loop of specification change and re – estimation.

Risk in finalizing specification (part 3)

History of Finalizing Specification

3 rd Stage

- **Now developers looks for methodology that minimize the specification changes.**

  - **Engineers began to think that a part of responsibility of specification changes is on the developer's side.**
  - **More prototyping is applied ("throw – away" is desirable)**
  - **JARR (Joint Application Requirement Resolution)**