Waltzing with Bears: Managing Risk on Software Projects

https://www.amazon.com/Waltzing-Bears-Managing-Software-Projects/dp/0932633609

Authors: Tom DeMarco, Timothy Lister
Published: March 2003

Overview

Part I: Why bother to do risk management?

- Risk is always there, so manage it instead of ignoring it.
- Risk is a possible future event that will lead to an undesirable outcome (problem). There are 5 steps to manage risk: discovery, analysis, planning, mitigation, monitoring.
- Unmanaged risks that materialize into problems are very costly, both to the company and to individuals.
- Good process alone isn't sufficient to mitigate risks.
- Risk management provides many benefits.

Part II: Why shouldn't we do risk management?

- Old management styles are incompatible with it. You'll likely fail if you're doing it by yourself.
- It's at odds with managing for success because you can't *manage away* risks; they're always there.
- Risk management is incompatible with culture if you tolerate failure but not uncertainty.
- Counting on lucky breaks is not risk management.
- You can ignore risks when the probability of materialization is small enough, you can't do anything about it should it happen, there are minimal consequences, or it's someone else's risk.

Part III: How shall we go about it?

- There's a 9-step prescription (updated to 18 steps later on) for discovering, managing, and monitoring risk. Figure out which ones you'll avoid, accept, mitigate, or evade. Compute your costs of exposure, containment, and mitigation.
- Quantify unknowns by coming up with windows of uncertainty based on past performance (or at worst, guesses).
- There are programs or Excel worksheets to run numbers for causal and aggregate risks. Modeling can also simulate these risks.
- Risk reserves are the time and money you set aside to contain risks.
- Try to think about risk like a probability distribution (graphs) rather than a single number.
- Discover risks by brainstorming catastrophes, working backward to scenarios, then doing a root cause analysis.
- Monitor for transition indicators (risk becoming a problem), always be discovering risks, collect data, and track how complete your project is.
- Common software risks: schedule flaws, requirements inflation, turnover, specification breakdown, under-performance.
- Start projects early.

Part IV: How much risk should our organization be willing to take?

- You can't just hand-wave the potential benefits. Without precision here, you can't quantify ROI or manage risk effectively.
- Benefits can be shown using uncertainty diagrams.
- Instead of looking at the aggregate cost/benefit (i.e., entire project), look at its components so that you can rank them accordingly.
- Compare the uncertainty of cost/schedule with the uncertainty of benefit to avoid death marches.
- There's an 18-step process for managing risk and keeping information flowing.

Part V: How do we know if it's working?

1. You have a risk register with common software project risks, and there are triggers.
2. You have a process for continually discovering risks, and it's open to others.
3. You use uncertainty diagrams.
4. Projects have goals and estimates, and they are different numbers.
5. All risks are monitored for transitions happening.
6. Each risk has contingency and mitigation plans.
7. Exposure has been calculated for each risk.
8. Projects have quantified values, and the components are ranked on them.
9. You deliver incrementally.

http://www.systemsguild.com/riskology

Part I: Why

Running Toward Risk

- "If a project has no risks, don't do it." Without risk, there's likely little benefit either.
- "Companies that run away from risk and focus on what they can do well are ceding the field to their adversaries." For example, Merrill Lynch was about a decade late to the online trading capabilities created by Fidelity, Schwab, and E-Trade.
- Bob Charette's risk escalator
    - You and your competitors make up a set of downward escalators
    - Each company has to climb up (moving against the flow)
    - If you drop off, you're out of the game
    - Everyone (including new competitors) start halfway up (i.e., potentially ahead of you)
    - Whoever gets to the top of their escalator controls a lever that changes the speed of all escalators
- We sometimes adopt a "can do" attitude to ignore risks; staying positive means we refuse to look at the downside

Risk Management Is Product Management for Adults

- Children can be excused from thinking about what can go wrong in the world, but adults must consider those things to keep themselves and the children safe
- "Taking explicit note of bad things that can happen (risks) and planning for them accordingly is a mark of maturity."
- **Risk** - a possible future event that will lead to an undesirable outcome
- **Problem** - the risk materialized
- "Risk management is the process of thinking out corrective actions before a problem occurs, while it's still an abstraction. The opposite of risk management is *crisis management*, trying to figure out what to do about the problem after it happens."
- **Risk transition** - the triggering event that turns a risk into a problem
- **Transition indicator** - what you see that shows you the transition is happening
- **Mitigation** - steps you take to keep your options open and to make correction possible later
- Components
    - Risk discovery -- brainstorm and triage what could go wrong
    - Exposure analysis -- determine probability of materializing and potential impact
    - Contingency planning -- what you expect to do when the problem exists
    - Mitigation -- steps to take before transition so that contingency plans are effective
    - Ongoing transition monitoring -- track managed risks; look for materialization

Denver International Airport Reconsidered

- Scene: A new Denver airport was scheduled to open in October 1993, but ended up being $500M over budget and 2 years late. The blame was placed on the lateness of the software for the automated baggage handling system, because the airport was designed such that, without this system, the airport could not function.
- "Even the most perfect construction process can’t remove uncertainty from a complex systems development project. Where there is uncertainty, there is risk. Where there is risk, there needs to be a conscious and thoughtful effort to manage it."

- Instead of asking how the software build process is faulty, ask about risk management
    - Why couldn't the airport open without this software?
    - Why was this system on the critical path?
    - Were there no alternatives to moving baggage?
    - When the system was late, why couldn't the airport use alternatives?
    - Could the airport have been designed to support alternatives (because it wasn't)?
    - Couldn't have preparing for alternate designs happened sooner?
    - Wasn't system lateness seen as a risk?
    - Haven't software projects been late before?
    - Was there any history of similar systems implemented at other airports?
    - Were other systems studied?
    - Did airport management follow advice from others who tried this?
    - The software team warned you about lateness, so why did you ignore them?
- "Responsibility for risk management accrues to whichever party will have to pay the price for the risks that are ignored."

The Case for Risk Management

- Corporate culture typically makes "impossible promises" such as deadlines, and instead of pushing back, project managers comply and cross their fingers.
- "Project managers often tell us that their clients would never do *any* projects if they understood the downside." So they just conceal the risk, and gradually share the bad news a bit at a time as it occurs.
- "Your willingness to commit to a risky project is a direct function of how well you can logically conclude that the risks have been assessed, quantified, and confronted."
- Makes "can't do" thinking okay
- Sets projects up for success
    - Are things stretch goals or reasonable expectations?
    - "People have little heart for work that leads them from one failure to another. The cost in morale, burnout, and poor employee retention is substantial."
- Bounds uncertainty (with boundless uncertainty, people are risk averse or foolhardy -- neither is good)
- Provides minimum-cost downside protection (risk reserve - time and money you may not need)
- Protects against invisible transfers of responsibility
- Maximizes opportunity for personal growth (no new directions = no growth)
- Protects management from getting blindsided (there's usually some warning before a problem occurs)
- Focuses attention where it's needed

Part II: Why Not

The Case Against Risk Management

- Risk management runs counter to certain (counterproductive) management styles, that are still practiced. Most of them are mechanical and are followed by people who don't know how to manage well: management by objectives, Parkinsonian scheduling (work expands to the time allotted to it), "culture of fear".
- Typical arguments against risk management
    - Our stakeholders can't face the risk
        - But… lying to people isn't sustainable
    - There's too much uncertainty
        - But… just ignoring uncertainty leads to the illusion of control
    - Uncertainty excuses poor performance
        - But… use goals to have people strive for best performance
    - "Manage for success" is better
        - But… you can't manage risk out of your project; it's inherent
    - Not enough data to manage risk effectively
        - But… you probably have some for common risks
    - Risk management in isolation is dangerous
        - CORRECT! You need support of others, otherwise you have a culture where people learn that it's more important to promise big than to deliver.

The Onus of Uncertainty

- "If your corporate culture won't allow you to admit uncertainty, you can't do risk management." (a.k.a. "It's okay to be wrong, but not okay to be uncertain.")
- Selective myopia -- only focusing on the risks you want to see (instead of the catastrophic ones you don't want to think about)
- "Attack your nightmares, not your petty worries; to discover the risks that really matter to your project, trace backward from effect to cause. Watch for oncoming trains."

Luck

- Things that make a risk not worth managing (a.k.a. we get hit by an asteroid)
    - Probability of materialization is small enough to ignore
    - Should it materialize, what we're building is now irrelevant
    - Risk has minimal consequences and requires no mitigation
    - Somebody else's risk
- If your project planning involves catching lucky breaks, this is not risk management.
- "Projects that start off as personal challenges seldom have their risks managed sensibly. They depend instead on luck."
- "When you take every chance in order to win, you may raise the consequences of losing, far beyond where they need to be." (e.g., in the lead for an Indy 500 race, almost out of gas, choose to not make the pit stop and end up running out of gas and losing the race)

Part III: How

Quantifying Uncertainty

- If there's no chance of hitting the planned deadline, you can't just stop there. What's a date you'll likely have it done (even if it's way in the future), and what's a date you're 50% likely to have it done?

- Risk diagrams help you quantify uncertainty.
- "The pathology of setting a deadline to the earliest articulable date essentially guarantees the schedule will be missed."
- "The test of a grown-up organization is that managers at all levels learn to live with commitments that have explicitly stated uncertainty bounds."
- **Nano-percent date (N)** -- the first date with a non-zero likelihood of delivery (any date prior has 0% chance)
- As an industry, we are encouraged to keep uncertainty windows small.
- Past performance can help you compute the window.
- Some studies show that the window size is 150-200% of N (e.g., scheduled to deliver on month 25? You may have to go out to month 75 to be totally sure)

Mechanics of Risk Management

- "Most software project managers do a reasonable job at predicting the tasks that *have to be done* and a poor job of predicting the tasks that *might have to be done*."
- Use the retrospectives from previous projects/releases as a starting point for risk of future projects/releases.
- What you can do about risk
  - **Avoid** -- don't do a project or part of the project that has risk
  - **Contain/accept** -- prepare (time and money) to pay for the risk should it materialize
  - **Mitigate** -- take steps to reduce containment cost
  - **Evade** -- this is basically crossing your fingers and being lucky
- "As a general rule, there are no contracts that successfully transfer *all* responsibility to a single party. If you are either client or contractor, expect to have to do some risk management."
- **risk exposure** = cost x probability
- "Assessing exposure is not a well-defined science. Your best guess about likely materialization may come from industry data, previous problem lists, the risk repository, or just a flat-out guess. Don't excuse yourself from this essential act just because any answer you come up with will never be demonstrably correct."

- **Showstopper** -- risk that if materialized would kill the project or even the company; these go higher up the org chart than the project, and at the project level these risks should be assumed not to exist
- **Risk reserve** -- time and money set aside to contain risks
- **Containment cost** -- time and money needed to address a materialized risk
- **Mitigation cost** -- time and money needed to reduce risk; should be offset by reduced containment cost, otherwise it's not worth it
- For each managed risk, there should be one or more indicators of materialization; watch closely for these happening.

Risk Management Prescription

1. Discover risks and write them down
2. Ensure typical software risks are written down too
3. For each risk…
    a. Assign a unique name and ID
    b. Find a transition indicator
    c. Estimate cost/schedule impact if materialized
    d. Estimate probability
    e. Calculate cost/schedule exposure
    f. Define contingency actions should it materialize
    g. Add mitigation actions to project plan
4. Designate showstoppers as project assumptions (move risk up)
5. Determine nano-percent date (N), assuming no risks materialize
6. Construct a risk diagram that involves N
7. Express commitments using risk diagrams with uncertainty windows
8. Monitor risks (execute contingency plans if needed)
9. Continue risk discovery throughout the project

- N is not a date you commit to; it's only an input into the process of finding something better to commit to
- N < goal date < scheduled delivery date
- If you fix the delivery date, then the uncertainty is about what will be delivered.
- Incremental implementation (e.g., Agile) helps.
- Make your risk register public so that other stakeholders can monitor and so that no one is surprised.

Back to Basics

- Project management requires answers to things like "when will you be done", "will the user accept our product", etc. If your answer is "I don't know", you've just identified an area of risk.
- **Uncertainty diagram** -- bar chart of outcomes on the x-axis, and probability of occurrence on the y-axis; you can also present this as a cumulative graph (i.e., PDF vs CDF for statistical distributions)
- **Risk** -- weighted pattern of possible outcomes and their associated consequences; this definition moves from thinking about risk numerically, and more like a probability distribution (graph)
- **Causal risk** -- risk pertaining to a component
- **Aggregate risk** -- risk for the project as a whole (fed by the causal risks)
- You can build risk models for causal/component risk and use those to model project risk.

Tools and Procedures

- If you have distributions of risk for certain variables, you can combine them with models and sampling. You can also get probabilities of certainties on a spectrum (e.g., how much time will this take where I can be 75% confident of hitting it?).
- See http://www.systemsguild.com/riskology for some tools and templates

Core Risks of Software Projects

- Core Risk #1 - Schedule Flaw
  - Tendency to misjudge the size of the product to be built
  - Upper management rarely blames the schedule; they blame the makers of the schedule
  - For typical projects, there's a 50% chance your schedule will overrun by 30%.
- Core Risk #2 - Requirements Inflation
  - "One thing you can be sure of is that the domain is not going to remain static during the building process."
  - For typical projects, there's a 50% chance your schedule will overrun by 7% because of this.
- Core Risk #3 - Turnover
  - Need to know average turnover percentage per year for technical staff, and the ramp-up time for a replacement (number of months to effectively replace someone who left).
  - For typical projects, there's a 50% chance your schedule will overrun by 4% because of this.
- Core Risk #4 - Specification Breakdown
  - This is where people can't agree on what product to build.
  - "While it's possible to *specify* a product ambiguously, it is not possible to *build* a product ambiguously."
- Core Risk #5 - Under-performance
  - Because there is a spectrum across the team, and other risks have a stronger impact than this one, the likelihood of this impacting schedule is very low.

A Defined Process for Risk Discovery

- Typical corporate culture discourages people speaking negatively about a project, only wants to hear about issues so long as they have solutions in hand, or makes people who speak up about certain issues become responsible for said issues.
- Brainstorm catastrophic outcomes, work backward to find the situations that could lead to those outcomes, and then work backward to identify the root causes of those situations (which are now your risks).
- Risk discovery doesn't happen just once; it's a continual process.
- **Step 1: Catastrophe Brainstorm**
  - Frame it in terms of a nightmare
  - What would the headlines be in the news?
  - What are the best dreams, then invert
  - How could the project fail and be no one's fault
  - How could the project fail and be our fault?
  - What does partial failure look like?
- **Step 2: Scenario Building**
  - Spell out the scenarios that lead to catastrophes
  - Attach a probability as well
- **Step 3: Root Cause Analysis**

- From Barry Boehm's WinWin Spiral Process Model… Have all stakeholders define win conditions for the project. Where you have two "wins" that are in conflict, you have risk.

Risk Management Dynamics

- Basic activities
  - Monitor transition indicators
  - Ongoing risk discovery
  - Collect data for the risk repository
  - Track closure metrics
- **Closure metric** -- how much of the project is done (e.g., **earned value running** (EVR))

Incrementalism for Risk Mitigation

- This is basically iterative development where you deliver subsets of functionality.
- Reactive incrementalism -- let the devs choose which subsets to work on with no priorities/judgment from management (avoid this)
- Proactive incrementalism -- careful plan based on value delivery and confirmation of risk hypotheses; rank-order the features to show that (1) all items are not equally important, and (2) piling on features beyond what you commit to is not allowed.
- "Those parts of the system that depend on pulling off technical wonders should be pushed into the early versions."
- **Incremental delivery plan** -- formal correlation between three artifacts
  - Design blueprint (modules and their relationship)
  - Work breakdown structure (tasks and dependencies)
  - Acceptance tests
- **Glide path** -- amount of EVR demonstrated complete per unit of time; deviations from this are a sure sign of risk manifestation

The Ultimate Risk Mitigation Strategy

- Starting early is a very effective way to mitigate the risk of being late
- Gutsy management is willing to take on a bit of risk to come out with an enhanced position if risk-taking pays off.
- "And projects that start too late are a sign of missing vision and courage at the top of the managerial ladder."

Part IV: How Much

Value Quantification

- ROI = (value - cost) / cost
- "Today, instead of building systems that offset cost, we more often embark on projects intended to improve our position in a market. These market-enhancing systems are much more complicated to justify." (e.g., we just have to have it, we need it to remain competitive)
- With vague benefit analysis, you can't do risk management. Both the costs and the benefits need to be quantified.
- There are various reasons why people try to make excuses for not doing benefits analysis; a strong one, is that it dilutes accountability.
- 5 elements of benefit calculation
    - If developers declare costs/schedules, stakeholders must declare benefits
    - If developers declare uncertainty, stakeholders must declare as well
    - Stakeholders break the system into components, each with cost-benefit analysis
    - Management uses cost-benefit analyses to justify the project
    - Management does a before/after comparison to see how they did

Value Is Uncertain, Too

- Don't just have a single (usually most optimistic) benefit value; use an uncertainty diagram.
- The **market window** is the most common excuse for avoiding careful benefit projections -- just start working now or the window will close! (Usually that date is difficult to achieve anyway.)

Sensitivity Analysis

- The only reason a project should get funded is that there's some value, and that the resulting product will deliver that value.
- Value is often only measured for the entire project; however, parts of the system likely have more/less value than others.
- Aim to compute a value/cost ratio for each piece, then show them in a tree/dependency diagram to make the data more visible.
- There is a *non-linear* relationship between the size of the project and the effort required to build it. It's cheaper to build less.

Value Offsets Risk

- "When the stakes are high, it's worth running even serious risks. When the stakes are low, almost no risk should be tolerable."
- Death-marches -- everyone must work harder/longer to deliver the project. But if the project requires this (because it's so important), why didn't the company spend the effort required to do it correctly? Is it really worth people's individual lives for the company's benefit? Most such scenarios don't end up well, and no one feels the better for it.
- Your project must balance the distribution of risk against the distribution of value.

Refining the Risk Management Prescription

1. Discover risks and write them down
2. Ensure typical software risks are written down too
3. For each risk…
    a. Assign a unique name and ID
    b. Find a transition indicator
    c. Estimate cost/schedule impact if materialized
    d. Estimate probability
    e. Calculate cost/schedule exposure
    f. Define contingency actions should it materialize
    g. Add mitigation actions to project plan
4. Designate showstoppers as project assumptions (move risk up)
5. Determine nano-percent date (N), assuming no risks materialize
6. Use models (like RISKOLOGY) to parameterize risk (this involves N)
7. Express commitments using risk diagrams with uncertainty windows; convince others based on the extent of your modeling
8. Create a work breakdown structure with relative weights of effort (these factor into earned value running metrics)
9. Get commitment on the architecture (data in, data out); treat this as a milestone and do not proceed without it
10. Do a complete design partition before any development
11. Revisit the WBS to re-estimate based on design
12. Assess value just as you assess cost
13. Put requirements in rank order based on net value and technical risk
14. Create an incremental delivery plan
15. Create a final acceptance test (i.e., definition of done)
16. Graph earned value running vs. planned delivery date of each increment; update as versions are released
17. Monitor all risks and execute plans as needed; monitor earned value running glide path
18. Keep discovering risks

Part V: Whether Not

Test for Risk Management

- Corporate culture gives lip service to saying a company does risk management. For example…
  - Thinking short-term
  - Needing to appear in control
  - Having a can-do attitude
  - Avoiding the appearance you aren't certain
  - Using political power

If you can get the first 6 of these, you're probably okay:

1. You have a risk register with common software project risks, and there are triggers.
2. You have a process for continually discovering risks, and it's open to others.
3. You use uncertainty diagrams.
4. Projects have goals and estimates, and they are different numbers.
5. All risks are monitored for transitions happening.
6. Each risk has a contingency and mitigation plans.
7. Exposure has been calculated for each risk.
8. Projects have quantified values, and the components are ranked on them.
9. You deliver incrementally.