What to do when things go wrong: Fixing train wrecks in progress

Scott Berkun
www.scottberkun.com

(c) Copyright 2004, 2005, Scott Berkun - www.scottberkun.com
Hi. I’m Scott.

- IE 1.0 -> 5.0, Windows, MSN
- www.scottberkun.com (uiweb.com)
- Consultant / Teacher / Writer
- The Book: “The art of project management”, O’Reilly 2005
The Art of project management

Table of Contents

1. A brief history and why you should care
2. The truth about schedules
3. How to figure out what to do
4. Visions
5. Where ideas come from
6. What to do with ideas
7. Specifications
8. Making decisions

1. Communication and relationships
2. How not to annoy people
3. When things go wrong
4. Leadership and trust
5. How to make things happen
6. Mid-game strategy
7. End-game strategy
8. Power and Politics
# The Art of project management

## Table of Contents

1. A brief history and why you should care
2. The truth about schedules
3. How to figure out what to do
4. Visions
5. Where ideas come from
6. What to do with ideas
7. Specifications
8. Making decisions

1. Communication and relationships
2. How not to annoy people
3. When things go wrong
4. Leadership and trust
5. How to make things happen
6. Mid-game strategy
7. End-game strategy
8. Power and Politics
Agenda

• Important announcement
• Basic rule of fatal problems
• Good teams vs. lousy teams
• Detection
• Assessment
• Action
• The rough guide
Important Announcement

- No frills book tour beer social
- Wednesday, May 11th, 7pm (tomorrow)
- Faultline Brewery, Sunnyvale
- http://www.faultlinebrewing.com/

- All are welcome
- Free books (while they last)
- Free beer (while my budget lasts)
Basic rules
The rule of fatal problems

By the time it’s obvious there is a fatal problem, it’s too late to do much about it.


The wise catch problems well before they are fatal.
Train wreck defined

• We have big problems
  – We know we won’t meet goals
  – No one is happy / Everyone is frustrated
  – Things keep changing, but there is little progress

• We don’t know if we’ll be able to solve them
• We don’t agree on the existence of problems
• We don’t know who’s job it is to solve them
Design disaster defined

- Disconnect between the “design” and what’s being built
- No one knows what the “design” is
- No one knows what the goals are
- There are competing designs being built simultaneously, and unintentionally, by different people

- The design has no possibility of satisfying goals
  - Customer / Technology / Business
  - (People with different goals will define disasters differently)
Good teams vs. Mediocre teams

• Good teams:
  – Avoid many problems and rat holes
  – Are good at recognizing/communicating issues
  – Are good at using each other to help solve its problems
  – Teach each other how to find and resolve problems
  – Make mistakes, but are encouraged to learn (not hide)

• One team’s train wreck is another team’s good day
Typical train wreck problems

- Disconnect between:
  - customers and team
  - business / engineering / design organizations
  - management and team
  - the team and reality

- Confused requirements
- Bad core architecture decisions
- Bad core UI design decisions
- Management is ineffective at leading team
- Rampant incompetence and stupidity
Detection
Toolkit for detection

- Knowledge of common kinds of failures
- Ability to compare to previous experiences
- Motivation and time to look for early symptoms
- Culture that listens to whistleblowers

Team processes should be constructed to:

- Make it easy to detect problems with the train
- Make it easy to fix problems
- Speed team up, not slow it down

- Lessons from the train yard: Measurement & Maintenance
The first step is admission

- Someone says: “We may have a problem”

- Who responds to warning flags of problems?
  - If no-one, the problem grows.
  - The lie of “We don’t have time” vs. prioritization
  - People stop telling you about problems.
  - Denial and fear spread. “Hot potato” issues.

- If you, or someone else with power..
  - Acknowledge problems, there is hope.
  - encourage others to look, there is a chance.
  - reward problem preventers, finders & solvers, there is confidence.
Assessment
Assessing problems

• What are the symptoms?
• What are the causes?
• When will it surface? How much damage?
• Do key players admit to the problem? Agree on priorities?

• If you prioritize the problems differently than others, then your first job is to change their minds
  – E.g. UX problems vs. other kinds of problems
  – Remember: you never want to fight a priority war during an emergency
Action
When was the last time you said “This problem is my responsibility”? 
Typical choices, strategic

- Change the train’s direction
- Slow the train
- Stop the train
- Crash, and admit failure (learn)
- Crash, and pretend you meant to crash
Typical choices, tactical

- **Add resources**
  - Invest more of your own time to fix the problem
  - Hire someone with expertise in diagnosing / fixing the problem
  - Increase the schedule

- **Cut something**
  - Give the team the resources to resolve the problem
  - Eliminate a requirement
  - Postpone a feature

- **Change something**
  - Modify how work gets done
  - Replace key player(s)
Figure 1. The Satir Change Model

From Quality Software Management, by Gerald Weinberg
Tactics for day 1

- What things can possibly go wrong?
- What things are likely to go wrong?
- What can we do now to prevent them?
- What kinds of detectors (canaries) do we need?

- This is called risk analysis / risk management
- Abstract: Your insurance premium is not a made up #
Tactics for mid-project

• Inspect for probable problems (Vaccinate)
• Ask team to make “risk lists” weekly / monthly
  – What are the 3 most likely things to go wrong next week?
  – What is the worst possible thing? (regardless of likelihood)
  – What can you / I / the team do to prevent these things?

• Make it part of 1-on-1s, team meetings.
• Make it fun and quick – don’t fixate or obsess.
• (Unless you uncover high probability issues)
Tactics for mid-project

• Re-prioritize around identified problems
• Get the team to agree to changed priority
  – “Be here now” – forget sunk costs & emotions.
  – If we do nothing, here are the consequences
  – To prevent this from getting worse, we need X
  – To make this better, we need Y
Tactics for late project

- When the train is about to wreck.. few options
- Reset / Pause the project
  - E.g. Microsoft’s 2003 Security Push, product recalls
- Cut your loses
  - agree to reduced goals
  - Renegotiate with customers
- Pick one single battle to fight and make a stand
- Plan for .1, v2, Postmortem
The rough guide
The rough guide (Chapter 11)

1. Calm Down
2. Evaluate the problem
3. Calm down again
4. Get the right people in the room
5. Explore alternatives
6. Make the simplest plan
7. Execute
8. Debrief
Summary

- Know the problems probable for your project
- Make people aware of the warning signs
- Good teams vs. lousy teams
- Reward investigation / anticipation of future problems
- Strategy: Change direction / slow / stop / crash
- Tactics: Add, cut, change people or process
Questions? Thoughts?

The art of project management, O’Reilly, Available now

Faultline Brewery, 7pm tomorrow

scott@scottberkun.com
www.scottberkun.com
Agenda

• Fixing train wrecks in progress
• Q&A
Lessons

• Picture of train wreck
  – Train wrecks can’t be fixed – by the time you know you have one it’s too late
  – Similar to various kinds of illness
  – You fix train wrecks by diagnosing them early enough to do something about them
  – Medical model is better, but wreck is more fun

• The four opportunities
  – Phase 1: not fully committed yet. Everyone is happy and everything seems possible.
  – Phase 2: Some things are committed. Some people are unhappy and some things are going poorly but can possibly be ignored.
  – Phase 3: A few things are clearly broken, but nothing has exploded yet. So how bad can it be.
  – Phase 4: wreck imminent or already happening

• The rough guide for what to do when things go wrong

• Who are your cannaries (Picture of cannery in a coal mine). And how can you support them so they don’t have to die for you to get the message. When the cannary dies, it’s already too late.

• The idea of software development as death marches. Hard core. Self flagellation. Lance Armstrong. (Are you sure you’re not making things harder than they need to be for your own reasons?)
Things are good

Things are ok

We’re worried

We’re in crash position