Table of Contents

Premise

When exploring new ideas we often encounter new risks. This session looks at how Agile risk management needs to be lightweight, but responsive to known risks as well as previously unknown sources of risk. Building new things means encountering new things that go wrong. This session gives techniques that help identify risks early through faciltated estimation, survey crowdsourcing and historical data analysis. We then discuss how to prioritize risks based on a system impact approach and probabilistic simulation.

We do risky things all the time. Avoiding risk isn't a desirable outcome, there is value in risky ventures. We simply need to know what risks are WORTH taking. We need to take more “wise” risks than our competitors, and this session aims to define what “wise” means and how to identify them in your software development portfolio and projects.

Although risk management is thought of as boring and laborious, it need NOT BE! Agile risk management even when performed in a lightweigth fashion is key to avoiding un-intended losses - in time; in costs; in missed opportunities. If teams and organizations had a better grip on assessing their risk exposure, better decisions and fewer smashed expectation would follow.

This talk covers - Where did risk management go in Agile? Why traditional risk management doesn't work in Agile Because single point likelihood and impact estimates - we need a range based approach Because risks and delays compound and small delays propogate into large delays - we need a systems approach How to find the risks that matter most By asking the teams better risk identification questions and by using survey crowdsourcing By looking at history and measuring occurrence and impact, and forecasting future impact By using probabilistic techniques to weigh total project impact rather than isolated impact of any one risk How to quantify risk impacts and make informed risk decisions By financially quantifying things that go wrong - dollars talk By using real option and decision tree techniques to move risk decision making down to the team level

Learning Outcomes: Understand Agile risk management goals and wher current techniques often fail The problem with single point estimates of risk occurrence likelihood and impact for prioritization Training and calibrating teams on estimation, and how poor human instinct is on risk estimation without help How to elicit range estimates from teams and how to extrapolate the high end (tail risk - low probability but high impact) Learn the basics about modeling, computing and communicating risk probability and impact Learn how to put a dollar value on total risk impact through probabilistic techniques

Summary

- Content rating (0-no new ideas, 5 - a new ideas/approach, 9-new ideas): 9
- Style rating (0-average presentstion, 5 - my level, 9-I learned something about presenting): 5

See learning - this was excellent. Style was pretty good given nature of the subject.

Action / Learning

Excellent presentation.

It basically showed why, for development projects, the “impact X probability”-type analysis is not really useful (issue is that if the event occurs, there is an impact on date, and we should be modeling risk associated with the release date / scope as, and if there are 2 or 3 events that come in there is a nonlinear impact on the release date), that how big something is in terms of work has limited effect of modeling risk as issues such as dependency of work and over utilization of people have far higher impact on the plan, and showed an approach to model risk more accurately using, you guessed it, Monte Carlo simulation. I've attached the presentation but, trust me, the pictures don't really help with the understanding of the main message unless you have background. Happy to talk to you about this if you want to.

Presentation

From Sim Resources

risk_-_the_final_agile_enterprise_frontier_troy_magennis_.pdf

Notes

Risk How well we hit an expectation or plan

How you deal with it

Plans and risk are about options

Risk - Anything that causes actual outcome to be different than planned outcome Plans of course can change

Expectation is the root of all heartache - Shakespeare

Risk About surface expectations

Two factor Penalty of being late Ability to alter outcome

Low loss, low ability to change - cost driven High loss, fixed - risk driven High loss, flexible - staff driven

Different need for risk management

Risk 1 Root cause

If working in congestion , harder to predict

How much does item size play a role in lead time Utilization is driver May work if not closer where there is at 100% utilization

Therefore - Keep utilization below a certain threshold Keep people under utilized to deliver on date Have capacity to absorb

This is number 1 risk Happy idle people

Look at system level impediments Have more impact

Trad risk management works if you have repeat work eg build house

Risk table Works for a lot of events - based on average For software projects this isn't good indicate of risk - happens once

For us date effect of risk

Zero risk 1 risk comes true and effect by delay in schedule

Leads to 4 delays means at least even chance that there is delay in projects And there are a lot of delays

Manage risk And not related to estimates As delay more related risks

Monte Carlo Probabilistic forecast

If planning on averages Fail on average if nothing else goes wrong

Average doesn't capture the big changes

See chart for risk management Get chart to Andrew

Bring the risk back means better chance of releasing earlier

This is real risk management

What are we going to solve in risk

Maximum waste is when we start work on something that is not viability Need to track ability to start Then look at things that effect multiple teams Then single teams Then stories

Need to have meeting on this

Ten failed forecasting plan assumption (most sensitive) See questions to drive

Top 1 for your project is different based on project

Missed start date Dashboard - missed start date y/n Give duration, not date Keep history of start

Team not ready, no team Team in place. Working on something else

Partial team Forecast assumes full team If one dashboard project is delayed

Partial body staffing Skills you need Level or number

Missing skill sets

Amdahls law Speed up from parallelizing any problem will be limited by serial 8 times cost get 3 time improvement See chart Coordination cost Twice people / teams is not twice the throughput

Dependencies Best way visualize is chart Looking for number levels

Technical debt

Ship stoppers

Splitting Mistake in forecasting Original backlog versus change