Thoughts on Software Architecture #2

Estimation, Monte Carlo Forecasting, Top 10 technology trends for 2020, Code Reviews at Google and Microsoft, Quantum Supremacy

Kai Niklas · Following
Nov 1, 2019 · 5 min read

Modern Software Architecture
Hand-picked posts on Modern Software Architecture.
Following
20

Thought of the Week

Software estimation is hard. Estimating with different people is even harder.

Business sponsors, customers, users, etc. are always keen to know more about their products or projects. Especially, they are interested in when things are ready. With estimations, we are able to answer these questions (more or less accurate). And with more data and experience forecasts are getting better over time.

As a developer or software architect we often estimate. And today, we talk about the soft side of the problem (people), which is from my point of view more problematic.

Scenario #1: Let's assume, we know quite well how long it will take. Then a manager, client or business sponsor comes into the room and claims: "Not acceptable. Think again. We need to deliver faster." If not handled properly, we will run into an uncomfortable situation, where we negotiate the estimate. We will somehow say, that we can deliver faster. But in most cases, this won't work out well.

The following video gives a great overview what to do in such a situations: "Estimates, Targets and Commitments" by Steve McConnell.

Scenario #2: Let's assume we are within a Scrum team with a stable velocity. What we claim to deliver, we often hit. BTW: High predictability is great! Then a manager, Product Owner, Scrum Master or someone else, enters the room and demands: "We need to go faster. With more experience, velocity has to increase." What may happen in this scenario?

1. The team may start **faking**. Estimates will go up artificially, because of Goodhart's law: The team will be faster, unfortunately on paper only.
2. The team may **burn out**. The team tries to go faster. It doing over-times. Eventually, less gets delivered, because people are/ get sick.
3. The team may build up **technical debt**. Shortcuts are taken which are not cleaned up anymore.
4. The team may invest **less in discovery work**. The team will have less time to understand business needs and thus, delivers less business value. Maybe more *output*, but less *outcome* with business impact.

What can we do? Make people aware of the situation and consequences their request is causing. Software development is not like working in a factory, where the goal is to produce as many items as possible, in a given time-frame, with the least amount of money. Software development involves knowledge workers. And working with knowledge is different. Outcome is achieved differently.

Resources to learn more:

- 17 Theses on Software Estimation
- Velocity should be renamed "future tech debt"
- Faking agile metrics

Tool / Method of the Week

▶ Forecasting in Scrum with Monte Carlo Simulation

Complementing the above thought process: More and more people are working in an agile way, often using Scrum. Business Owner, clients or customers are often interested in forecasts and road-maps of when they can expect specific new features. Based on historic data it is possible to create forecasts. Most often, due to a lack of knowledge, naive approaches are used considering only the mean time to finish user stories.

Unfortunately, user stories are sometimes done faster, sometimes slower. Also considering story point estimations does not necessarily help, as blockers may appear and screw up numbers.

With a Monte Carlo simulation such deviation can be taken into account. Further, a likelihood can be specified to define the accuracy of the forecast, e.g., 95% or 6-sigma (99,99966%), depending on the risk level you want to take.

The following example is taken from my book "Become a better Software Architect." If you want to learn more and get detailed explanations, you can get a digital copy on leanpub or a hardcover on amazon.

- Simple forecasting in Scrum with Monte Carlo Simulation

Historic Data (more data or more accurate results)																			
Sprint	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Done User Stories	7																		
Velocity in Stories	7	6	10	8	9	7													
Simulation (Input)																			
Done	50																		
Velocity in Stories	10																		
Simulation (Forecast)																			
Average	8																		
95% Quant.	10																		

Usage Note:

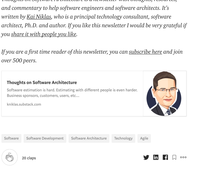
- **Provide historic data**: How many stories have been done?
- **Provide input for simulation**: How many stories are planned?
- **Read simulation results**: How many iterations are required with a likelihood of 95%?

Top Resources

▶ The Ultimate Code Review Blog Post Series
(by Dr. Michaela Greiler)

- Code Reviews at Microsoft
- Code review pitfalls: learn which problems slow your team down
- Proven Code Review Best Practices
- How to give great feedback
- Code Reviews at Google
- Code Review Checklist
- Don't criticize my code

▶ Gartner's top 10 technology trends for 2020



▶ Hello quantum world! Google publishes landmark quantum supremacy claim

The company says that its quantum computer is the first to perform a calculation that would be practically impossible for a classical machine.

Researchers outside Google are already trying to improve on the classical algorithms used to tackle the problem, in hopes of bringing down the firm's 10,000-year estimate. IBM, a rival to Google in building the world's best quantum computers, reported in a preprint on 21 October that the problem could be solved in just 2.5 days using a different classical technique.

▶ Analyzing the Pluralsight Tech Index: Trends in Software Dev, IT, Data and Security

What's impressive is the constant improvements to JavaScript that enable developers to continue to innovate and thrive in the web application world.

Languages like Python, C and C++ are on the rise, but the recent rise in machine learning and data analysis work is intersecting with these languages in interesting ways.

Wait, what is this?

Thoughts on Software Architecture is a newsletter with thoughts, resources, and commentary to help software engineers and software architects. It's written by Kai Niklas, who is a principal technology consultant, software architect, Ph.D. and author. If you like this newsletter I would be very grateful if you share it with people you like.

If you are a first time reader of this newsletter, you can subscribe here and join over 500 peers.

Thoughts on Software Architecture
Software estimation is hard. Estimating with different people is even harder. Business sponsors, customers, users, etc...
kniklas.substack.com

Software · Software Development · Software Architecture · Technology · Agile
20 claps

WRITTEN BY
Kai Niklas · Following
Software Architect | Software Engineer | Open-Minded | Perpetual | PhD | Innovation Principal Consultant | https://twitter.com/SoftwareArchitect.com

Modern Software Architecture
Hand-picked posts on Modern Software Architecture.
Following
Write the first response

Discover Medium
Welcome to a place where words matter. On Medium, smart voices and original ideas take center stage - with no ads in sight. Watch

Make Medium yours
Follow all the topics you care about, and we'll deliver the best stories for you to your homepage and inbox. Explore

Explore your membership
Thank you for being a member of Medium. You get unlimited access to insightful stories from amazing thinkers and storytellers. Browse

Medium
About · Help · Legal