40.317 Lecture 12

Dr. Jon Freeman

25 June 2020

Slido Event #K750

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Agenda

- Blockchains, part 2 of 2
  - Challenges
  - Alternatives to Proof of Work
  - Next-generation blockchains
- Hedera Hashgraph
- Concluding thoughts about design

Blockchain Challenges

In no particular order:

- Environmental
- Governance
- Performance
- Scalability
- Security
- … and Relevance, a *consequence* of the first five challenges

Challenges: Environmental

BTC consumes massive amounts of energy (as does ETH), and its mining hardware generates massive amounts of e-waste.

The problem is BTC's use of Proof of Work to obtain consensus.

PoW does have some compelling uses: preventing spam emails, for instance.

Challenges: Governance

It is extremely difficult to introduce improvements, a.k.a. hard forks.

- For security though, this is a strength!
- ETH has managed to introduce several, with great effort; their biggest challenge so far, ETH 2.0, is still to come

Blockchain developers tend to value freedom and de-centralisation, so establishing coordinated decision-making processes is a social & cultural challenge.

Challenges: Governance

- The most useful blockchains implement a standard.
- But the members of an entire ecosystem must *invent*, *approve*, and *adopt* it…
- …and they are usually competitors.
- Which is harder to achieve and offers more benefits: using a blockchain, or agreeing an industry-wide standard?
  - https://en.wikipedia.org/wiki/ISO_20022
  - A case study: Lygon

Challenges: Performance

In practice, BTC performs about 5 transactions per second on average.

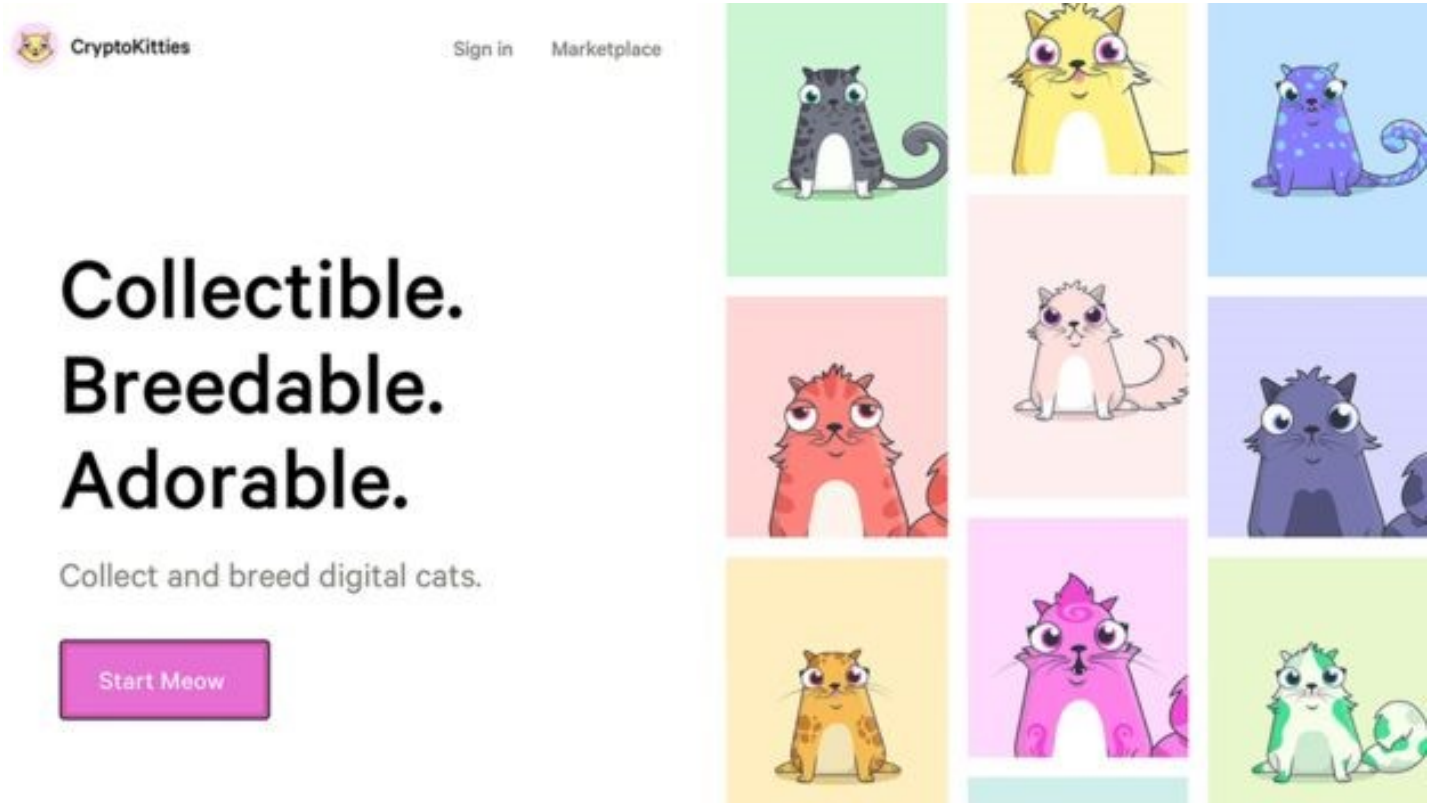
- Confirmation times are volatile as well as slow.

ETH performs about 15 per second.

For comparison, Visa performs about 1700 per second.

Challenges: Scalability

c.f. CryptoKitties, launched in late 2017.



Challenges: Scalability

- BTC's theoretical maximum is 27 transactions per second (on average).
- In its current form, ETH can perform at most 30 per second.
- Visa claim to be able to handle up to 24000 per second.

Throughput is primarily a consequence of block size and block creation time. Slow creation times are a consequence of PoW.

  - PoW *needs* to be slow for BTC, to ensure its chain growth is orderly!

Challenges: Security

- The various aspects of key management
- Selfish mining
- The 51% attack – many smaller PoW cryptos are extremely vulnerable to it
  - This actually happened in January 2019 to Ethereum Classic
  - BTC is less of a concern, *but* 10 mining pools have most of its hashing power

- PoW alternatives, discussed shortly, have other security concerns

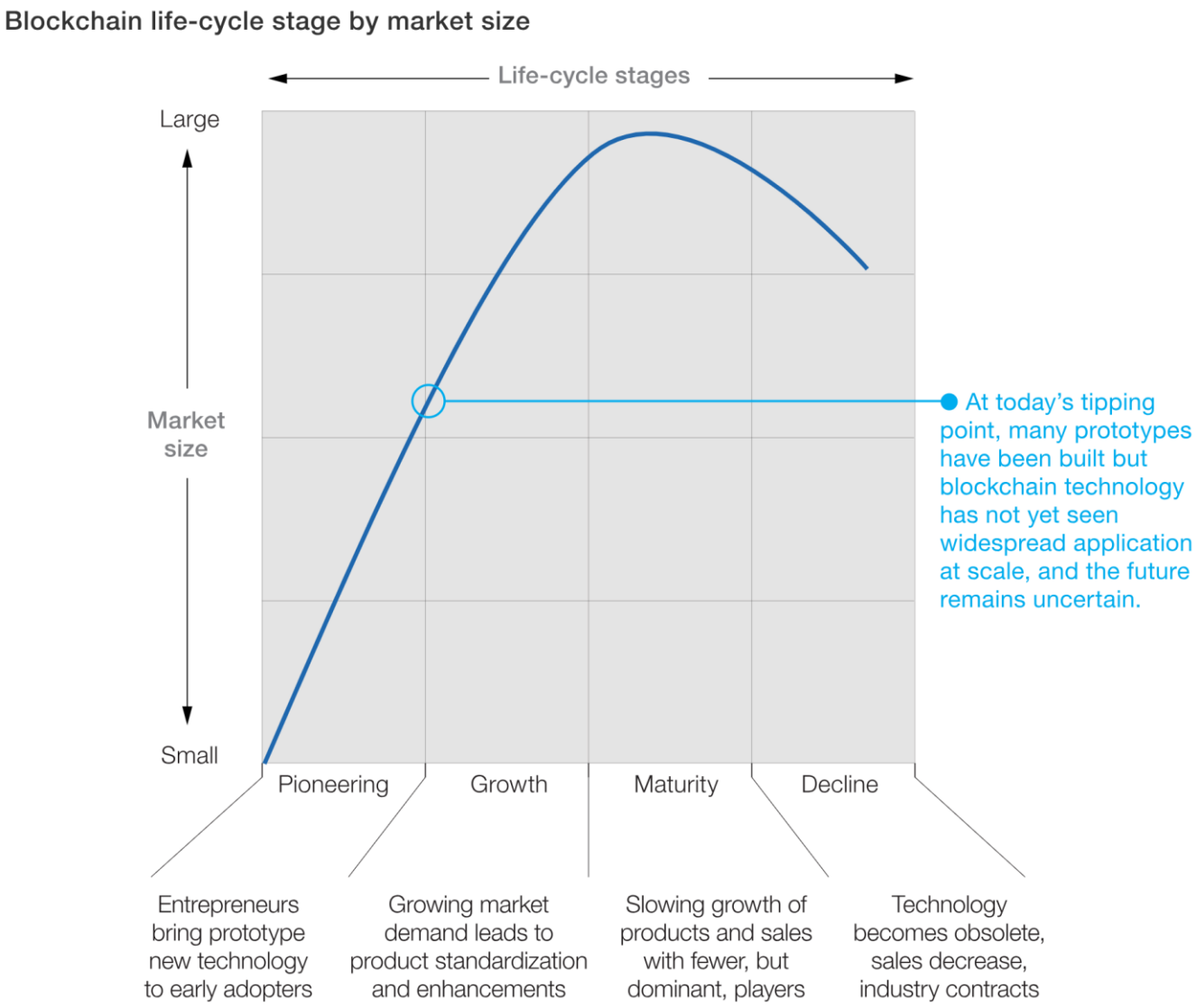
Challenges: Security

A system which employs a blockchain is not necessarily secure from end to end!

A good recent example was MIT's February 2020 security analysis of Voatz

- (Microsoft's approach to Internet voting, via ElectionGuard, looks more promising)

Challenges: Relevance



A BETTER WORLD BY DESIGN.

Challenges: Relevance

Per McKinsey:

- "[Our] work with financial services leaders over the past two years suggests those at the blockchain 'coalface' have begun to have doubts."
- "The *coopetition paradox* applied; few companies had the appetite to lead development of a utility that would benefit the entire industry."
- Incumbents are not standing still, e.g. SWIFT's gpi for cross-border payments.

Does Blockchain Have a “Killer App”?

- Crypto: BTC is underused; a March 2019 analysis claimed 95% of all BTC trading volume is fake; SEC has still not approved BTC ETFs after many years.
- Asset-backed tokens: infrastructure – social constructs, standards, and incentives – must all come first.
- User identity: needed most for displaced persons; for the rest of us, “[it] can be improved but it simply isn’t broken.”

Relevance, continued

The (limited) use case for blockchain *in its original form*:

Specific, well-defined, complex applications requiring transparency and tamper-resistance in preference to speed.

E.g., Visa's blockchain-based Visa B2B Connect.

Blockchain's Meta-challenge

In general, blockchain designers find themselves in a trilemma: they need to achieve the following three goals, but in practice can achieve at most two:

- Security
- Decentralisation
- Scalability

Popular Blockchain Platforms

Not a complete list, and subject to change.

- Ethereum
  - HydraChain, an Ethereum extension
- Hyperledger Fabric
- IBM Blockchain
- BigChainDB
- Openchain
- R3 Corda
- MultiChain

Alternatives to PoW

We will look at the following:

- Proof of Stake
- Proof of Burn
- Proof of Capacity
- Proof of Elapsed Time

Various next-generation blockchains have introduced others.

The above approaches are sometimes combined, e.g. an initial PoW phase followed by PoS.

Proof of Stake

The next block's creator is chosen at random, but weighted by # of coins *at stake* (and sometimes by coin age as well).
Its main challenges:

- Fair initial distribution
- Monopolisation
- 51% attack
- "Nothing at Stake" (NoS), i.e. nothing to discourage most nodes from signing 2 out of 2 competing chains

ETH has been switching from PoW to PoS.

Proof of Burn

Miners “burn” coins by sending them to an inaccessible address.

Miners which burn more coins are more likely to be chosen to add the next block.

Can use to replace one coin with another: burn the old, issue rewards in the new.

Challenges: similar to PoS, e.g. monopolisation.

Proof of Capacity

Introduced in the Burst blockchain.

Similar to PoW, but uses disk space instead of computation, hence greener and less vulnerable to specialised mining hardware.

Miners fill up the free space on their hard drives with pre-generated chunks of data containing all the computations necessary to forge blocks.

Proof of Elapsed Time

Used in Hyperledger Sawtooth.

Everyone is given a random amount of time to wait; the first person to be done waiting gets to determine the next block.

Requires running trusted code in a protected environment, which in turn requires specialised hardware, namely Intel SGX.

Next-Generation Blockchains

Novel blockchains to be aware of *include*:

- IOTA
- Byteball
- Nano (formerly Railblocks)
- NEM – introduces "Proof of Importance"
- IOST – introduces "Proof of Believability"
- And Hedera Hashgraph, discussed next

Next-Generation Blockchains

What if blockchains as we have come to know them are already obsolete?

- Three years ago, we saw fragmentation of the cryptocurrency market.
- Since then, we have been seeing fragmentation of blockchain methodologies themselves.

Is there a completely different way to obtain trustless distributed consensus?

Hedera Hashgraph

- Developed in 2016.
- Forms a consensus using a hashgraph algorithm. Here is a short explanation.
- A *public* ledger, available to all via the Hedera Consensus Service.
  - You need an account, denominated in HBARs, to pay (small) transaction fees.
  - There are testnets as well as the mainnet.
- Hedera-supported SDKs are currently available for Java, JavaScript, and Go.

Concluding Thoughts

- Revisiting agile software development
- Rethinking our guiding metaphor
- Final advice

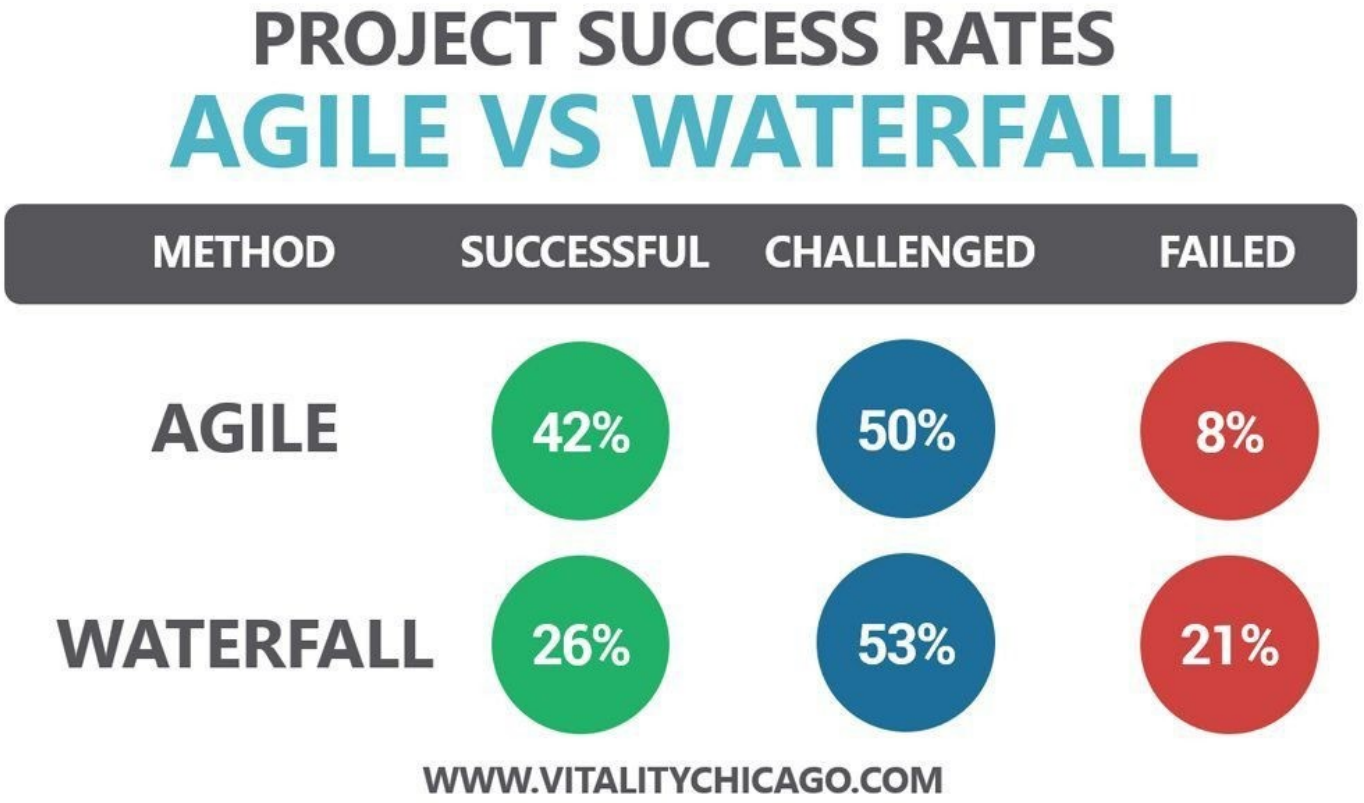
Thoughts about Agile

Let's remind ourselves of the principles of agile development, in 12 bullet points:

1. Satisfy the customer
2. Welcome change
3. Deliver frequently
4. Work together
5. Build projects
6. Face-to-face time
7. Measure of progress
8. Sustainable development
9. Continuous attention
10. Keep it simple
11. Organised teams
12. Reflect for effectiveness

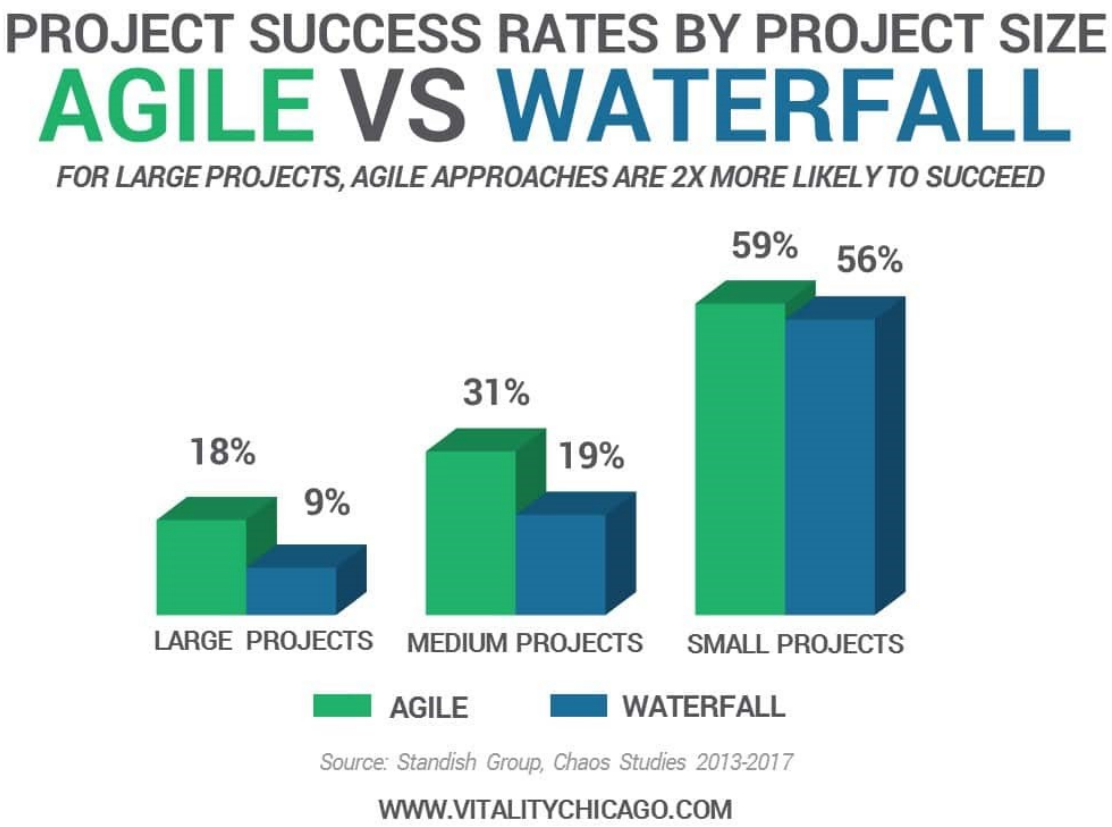
Thoughts about Agile, continued

What is your reaction to this 2019 data?



Thoughts about Agile, continued

And your reaction to this 2013-2017 data?



Thoughts about Agile, continued

The second graph shows a clear benefit of agile vs. waterfall for larger projects. This is most likely due to which agile practice(s)?

- Gall's Law?
- Ongoing stakeholder participation?
- Formal, periodic retrospectives?
- Developer empowerment?
- (Anything else?)

Thoughts about Agile, continued

Nevertheless, the first graph is worrisome and the second is troubling:

Digital computers have been around for *75 years*, and an 18% success rate for large projects is still the best we can do??

There must be an important consideration which most people are overlooking!

Thoughts about Agile, continued

Go back and re-read the 12 agile principles. *Where does the word "design" appear?*

So how can the use of agile guarantee a system will be well designed??

And regarding the second chart, isn't a good design critical for larger systems, and much less critical for smaller ones?

Thoughts about Agile, continued

Per Kurt Cagle:

- Agile is “merely a wish list, not a genuine methodology.”
- Its approach “is reductionist, and falls down when integrating components together.”
- It “decentralises responsibility too much.”
- It “does not distribute well.”

Thoughts about Agile, continued

Perhaps it will help to (re-)consider our guiding analogy for building software systems.

Building a software system is most like *which other activity* in *which other industry*?

- Civil engineers building a bridge?
- Surgeons performing an operation?
- What else?

Thoughts about Agile, continued

For your consideration, we now present The Studio Model:

Building software is most like creating a movie or television program.

The Studio Model has 12 principles as well, and they are:

12 Principles of The Studio Model

1. Vision is critical.
2. *Good design is an absolute requirement.*
3. Flexibility, not failure.
4. Change is exponentially expensive.
5. Redundancy matters.
6. Acknowledge cycles.
7. The customer is not the visionary.
8. Minimal viable products aren’t.
9. Complexity lives at the edges.
10. Stewardship plays a part.
11. Self-organizing groups don’t.
12. Recognise, reward and train up expertise.

Final Advice

Building software systems is not an art, and not a science, but something in between: a craft.

To build such systems well, you have to know what good software craftsmanship looks like.

Final Advice, continued

- Derive your project management methodology from the most suitable guiding analogy, namely the Studio Model.

- First design your system's data, then design its behaviour.
  - When designing its data, maintain a clear separation between mutable and immutable data.

Final Advice, continued

- Create your overall design by inventing barriers which partition your *problem* into independent services.
  - They can be, *but need not be*, implemented as actual [micro-]services.
- Within each service, invent more barriers to partition its *functionality* into independent packages.
- Last but not least, enforce these barriers as change requests arise over time.

Thank you

A BETTER WORLD BY DESIGN.

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN