

# **Part I**

## **Introduction**



# **Part II**

## **Risk**



# Coordination Risk

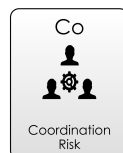
Coordination Risk is the risk that, a group of people (or processes), maybe with a similar Goal In Mind they can fail to coordinate on a way to meet this goal and end up making things worse. Coordination Risk is embodied in the phrase “Too Many Cooks Spoil The Broth”: more people, opinions or agents often make results worse.

As in Agency Risk, we are going to use the term *agent*, which refers to anything with agency<sup>1</sup> in a system to decide it’s own fate. That is, an agent has an Internal Model, and can take actions based on it. Here, we’re going to work on the assumption that the agents *are* working towards a common Goal, even though in reality it’s not always the case, as we saw in the section on Agency Risk.

In this section, we’ll first build up A Model Of Coordination Risk and what exactly coordination means and why we do it. Then, we’ll look at some classic Problems of Coordination. Then, we’re going to

---

<sup>1</sup><https://github.com/risk-first/website/wiki/Agency-Risk#software-processes-and-teams>



- Risks that a group of agents cannot work together in a mutually beneficial way, and their behaviour devolves into competition.

Figure 1.1: Coordination Risk

consider agency at several different levels (because of Scale Invariance)  
. We'll look at: - Team Decision Making, - Living Organisms, - Larger Organisations and the staff within them, - and Software Processes.

... and we'll consider how Coordination Risk is a problem at each scale.

But for now, let's crack on and examine where Coordination Risk comes from.

## 1.1 A Model Of Coordination Risk

Earlier, in Dependency Risk, we looked at various resources (time, money, people, events etc) and showed how we could Depend On Them, taking on risk. Here, however, we're looking at the situation where there is *competition for those dependencies*, that is, Scarcity Risk: other parties want to use them in a different way.

### Competition

The basic problem of Coordination Risk, then, is *competition*. Sometimes, competition is desirable (such as in sports and in markets), but sometimes competition is a waste and cooperation would be more efficient. Without coordination, we would deliberately or accidentally compete for the same Dependencies, which is wasteful.

Why is this wasteful?

One argument could come from Diminishing Returns<sup>2</sup>, which says that the earlier units of a resource (say, chocolate bars) give you more benefit than later ones.

We can see this in the graph below. Let's say A and B compete over a resource, of which there are 5 units available. For every extra A takes, B loses one. The X axis shows A's consumption of the resource, so the biggest benefit to A is in the consumption of the first unit.

As you can see, by *sharing*, it's possible that the *total benefit* is greater than it can be for either individual. But sharing requires coordination. Further, the more competitors involved, the *worse* a winner-take-all outcome is for total benefit.

---

<sup>2</sup>[https://en.wikipedia.org/wiki/Diminishing\\_returns](https://en.wikipedia.org/wiki/Diminishing_returns)

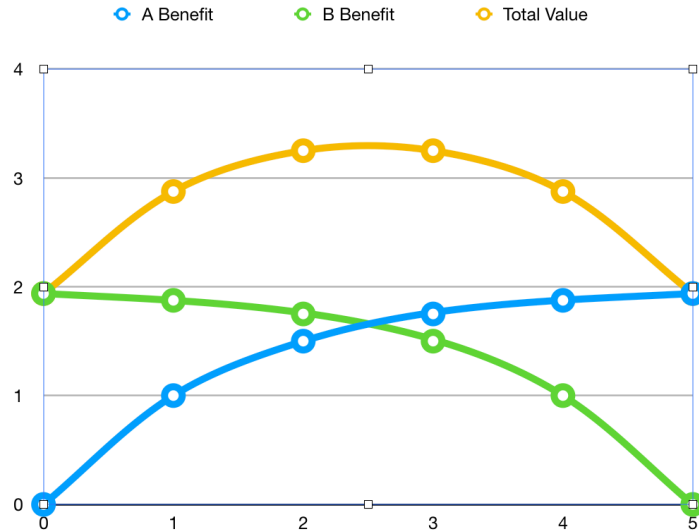


Figure 1.2: Sharing Resources. 5 units are available, and the X axis shows A's consumption of the resource. B gets whatever remains. Total benefit is maximised somewhere in the middle

Just two things are needed for competition to occur:

- Individual agents, trying to achieve Goals.
- Scarce Resources, which the agents want to use as Dependencies.

## Coordination via Communication

The only way that the agents can move away from competition towards coordination is via Communication, and this is where their coordination problems begin.

You might think, therefore, that this is just another type of Communication Risk problem, and that's often a part of it, but even with synchronized Internal Models, coordination risk can occur. Imagine the example of people all trying to madly leave a burning building. They all have the same information (the building is on fire). If they

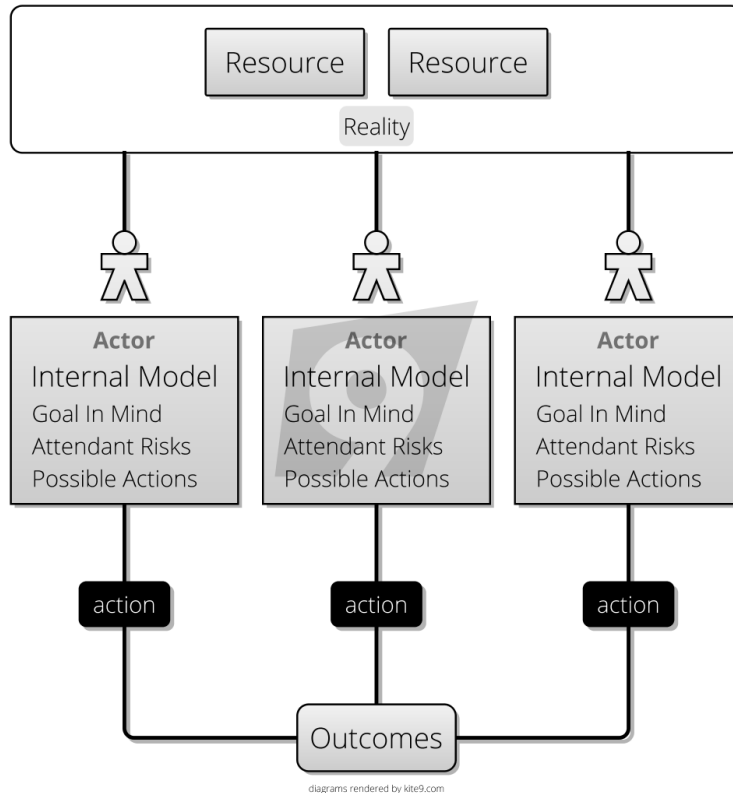


Figure 1.3: A model of competition: scarce resources, and individual agents competing for them.

coordinate, and leave in an orderly fashion, they might all get out. If they don't, and there's a scramble for the door, more people might die.

But commonly, Coordination Risk occurs where people have different ideas about how to achieve a goal, and they have different ideas because they have different evaluations of the Attendant Risk. As we saw in the section on Communication Risk, we can only hope to synchronize Internal Models if there are high-bandwidth Channels available for communication.



## 1.2 Problems Of Coordination

Let's unpack this idea, and review some classic problems of coordination, none of which can be addressed without good communication:

1. **Merging Data.** If you are familiar with the source code control system, Git<sup>3</sup>, you will know that this is a *distributed* version control system. That means that two or more people can propose changes to the same files without knowing about each other. This means that at some later time, Git then has to merge (or reconcile) these changes together. Git is very good at doing this automatically, but sometimes, different people can independently change the same lines of code and these will have to be merged manually. In this case, a human arbitrator “resolves” the difference, either by combining the two changes or picking a winner.
2. **Consensus.** Making group decisions (as in elections) is often decided by votes. But having a vote is a coordination issue, and requires that everyone has been told the rules:
  - Where will the vote be held?
  - How long do you provide for the vote?
  - What do you do about absentees?
  - What if people change their minds in the light of new information?
  - How do you ensure everyone has enough information to make a good decision?
3. **Factions.** Sometimes, it's hard to coordinate large groups at the same time, and “factions” can occur. That the world isn't a single big country is probably partly a testament to this: countries are frequently separated by geographic features that prevent the easy flow of communication (and force). We can also see this in distributed systems, with the “split brain”<sup>4</sup> problem. This is where a network of processes becomes disconnected (usually

---

<sup>3</sup><https://en.wikipedia.org/wiki/Git>

<sup>4</sup>[https://en.wikipedia.org/wiki/Split-brain\\_\(computing\)](https://en.wikipedia.org/wiki/Split-brain_(computing))

due to a network failure between data centers), and you end up with two, smaller networks with different knowledge. We'll address in more depth later.

4. Resource Allocation<sup>5</sup>: Ensuring that the right people are doing the right work, or the right resources are given to the right people is a coordination issue. On a grand scale, we have Logistics<sup>6</sup>, and Economic Systems<sup>7</sup>. On a small scale, the office's *room booking system* solves the coordination issue of who gets a meeting room using a first-come-first-served booking algorithm.
5. Deadlock<sup>8</sup>: Deadlock refers to a situation where, in an environment where multiple parallel processes are running, the processing stops and no-one can make progress because the resources each process needs are being reserved by another process. This is a specific issue in Resource Allocation, but it's one we're familiar with in the computer science industry. Compare with Gridlock<sup>9</sup>, where traffic can't move because other traffic is occupying the space it wants to move to already.
6. Race Conditions<sup>10</sup>: A race condition is where we can't be sure of the result of a calculation, because it is dependent on the ordering of events within a system. For example, two separate threads writing the same memory at the same time (one ignoring and over-writing the work of the other) is a race.
7. **Contention**: Where there is Scarcity Risk for a Dependency, we might want to make sure that everyone gets fair use of it, by taking turns, booking, queueing and so on. As we saw in Scarcity Risk, sometimes, this is handled for us by the Dependency itself. However if it isn't, it's the *users* of the dependency who'll need to coordinate to use the resource fairly, again, by communicating with each other.

---

<sup>5</sup>[https://en.wikipedia.org/wiki/Resource\\_allocation](https://en.wikipedia.org/wiki/Resource_allocation)

<sup>6</sup><https://en.wikipedia.org/wiki/Logistics>

<sup>7</sup>[https://en.wikipedia.org/wiki/Economic\\_system](https://en.wikipedia.org/wiki/Economic_system)

<sup>8</sup><https://en.wikipedia.org/wiki/Deadlock>

<sup>9</sup><https://en.wikipedia.org/wiki/Gridlock>

<sup>10</sup>[https://en.wikipedia.org/wiki/Race\\_condition](https://en.wikipedia.org/wiki/Race_condition)

## 1.3 Team Decision Making

Within a team, Coordination Risk is at it's core about resolving Internal Model conflicts in order that everyone can agree on a Goal In Mind and cooperate on getting it done. Therefore, Coordination Risk is worse on projects with more members, and worse in organizations with more staff.

If you are engaged in a solo project, do you suffer from Coordination Risk at all? Maybe: sometimes, you can feel "conflicted" about the best way to solve a problem. And weirdly, usually *not thinking about it* helps. Sleeping too. (Rich Hickey calls this "Hammock Driven Development"<sup>11</sup>). This is probably because, unbeknownst to you, your subconscious is furiously communicating internally, trying to resolve these conflicts itself, and will let you know when it's come to a resolution.

Vroom and Yetton<sup>12</sup> introduced a model of group decision making which delineated five different styles of decision making within a team. These are summarised in the table below (**AI**, **AII**, **CI**, **CII**, **GII**). To this, I have added a sixth (**UI**), which is the *uncoordinated* option, where everyone competes. In the accompanying diagrams I have adopted the following convention: - Thin lines with arrow-heads show a flow of *information*, either one-way or two-way. - Thick lines show a flow of *opinion*. - Boxes with corners are *decision makers*, whereas curved corners don't have a part in the decision.

At the top, you have the *least* consultative styles, and at the bottom, the *most*. At the top, decisions are made with just the leader's Internal Model but moving down, the Internal Models of the rest of the team are increasingly brought into play.

The decisions at the top are faster, but don't do much for mitigating **Coordination Risk**. The ones below take longer, (incurring Schedule Risk) but mitigate more **Coordination Risk**. Group decision-making inevitably involves everyone *learning*, and improving their Internal Models.

The trick is to be able to tell which approach is suitable at which time. Everyone is expected to make decisions *within their realm of ex-*

---

<sup>11</sup><https://www.youtube.com/watch?v=f84n5oFoZBc>

<sup>12</sup>[https://en.wikipedia.org/wiki/VroomYetton\\_decision\\_model](https://en.wikipedia.org/wiki/VroomYetton_decision_model)

Type	People Involved In Decision	Opinions	Channels Of Communication	Coordination Risk	Description
UI	1	1	0	Competition	<i>No Coordination</i>
AI	1	1	s (One message to each <b>subordinate</b> )	Maximum Coordination Risk	Autocratic, top-down
AII	1	1	2 × s (Messages from/to each <b>subordinate</b> )		Autocratic, with informal flow up.
CI	1	1 + s	> 2 × s		Individual Consultations
CII	1	1 + s	> s2		Group Consultation
GII	1 + s	1 + s	> s2	Maximum Communication Risk, Schedule Risk	Group Con-sultation with voting

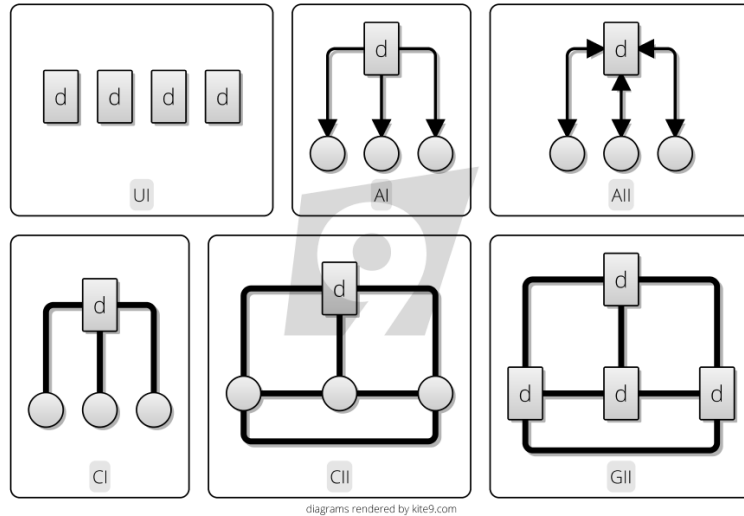


Figure 1.4: Vroom And Yetton Decision Making Styles. “d” indicates authority in making a decision. Thin lines with arrow-heads show information flow, whilst thick lines show *opinions* being passed around.

*pertise*: you can’t have developers continually calling meetings to discuss whether they should be using an Abstract Factory<sup>13</sup> or a Factory Method<sup>14</sup>, this would waste time. The critical question is therefore, “what’s the biggest risk?” - Is the Coordination Risk greater? Are we going to suffer Dead End Risk if the decision is made wrongly? What if people don’t agree with it? Poor leadership has an impact on Morale too. - Is the Schedule Risk greater? If you have a 1-hour meeting with eight people to decide a decision, that’s *one man day* gone right there: group decision making is *expensive*.

Hopefully, this model shows how *organisation* can reduce Coordination Risk. But, to make this work, we need more *communication*, and this has attendant complexity and time costs. So, we can draw this diagram of our move on the Risk Landscape:

<sup>13</sup>[https://en.wikipedia.org/wiki/Abstract\\_factory\\_pattern](https://en.wikipedia.org/wiki/Abstract_factory_pattern)

<sup>14</sup>[https://en.wikipedia.org/wiki/Factory\\_method\\_pattern](https://en.wikipedia.org/wiki/Factory_method_pattern)

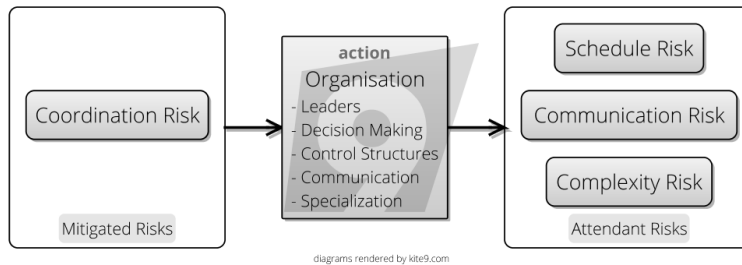


Figure 1.5: Coordination Risk traded for Complexity Risk, Schedule Risk and Communication Risk

## Staff As Agents

Staff in a team have a dual nature: they are **Agents** and **Resources** at the same time. The team depends on staff for their resource of *labour*, but they're also part of the decision making process of the team, because they have agency over their own actions.

Part of Coordination Risk is about trying to mitigate differences in Internal Models. So it's worth considering how varied people's models can be: - Different skill levels - Different experiences - Expertise in different areas - Preferences - Personalities

The job of harmonizing this on a project would seem to fall to the team leader, but actually people are self-organising to some extent. This process is called Team Development<sup>15</sup>:

"The forming–storming–norming–performing model of group development was first proposed by Bruce Tuckman in 1965, who said that these phases are all necessary and inevitable in order for the team to grow, face up to challenges, tackle problems, find solutions, plan work, and deliver results."  
- Tuckman's Stages Of Group Development, *Wikipedia*

Specifically, this describes a process whereby a new group will form and then be required to work together. In the process, they will have many *disputes*. Ideally, the group will resolve these disputes internally and emerge as a *Team*, with a common Goal In Mind.

<sup>15</sup>[https://en.wikipedia.org/wiki/Tuckman%27s\\_stages\\_of\\_group\\_development](https://en.wikipedia.org/wiki/Tuckman%27s_stages_of_group_development)

They can be encouraged with orthogonal practices such as team-building exercises<sup>16</sup> (generally, submitting everyone to extreme experiences in order to bond them together). With enough communication bandwidth and detente, a motivated team will self-organise code reviews, information exchange and improve their practices.

As described above, the job of Coordination is Resource Allocation, and so the skills of staff can potentially be looked at as resources to allocate. This means handling Coordination Risk issues like:

- People leaving, taking their Internal Models and expertise with them Key Man Risk.
- People requiring external training, to understand new tools and techniques Learning-Curve Risk.
- People being protective about their knowledge in order to protect their jobs Agency Risk.
- Where there are mixed ability levels, senior developers not helping juniors as it “slows them down”.
- People not getting on and not helping each other.

“As a rough rule, three programmers organised into a team can do only twice the work of a single programmer of the same ability - because of time spent on coordination problems.” - Gerald Weinberg, *The Psychology of Computer Programming*<sup>17</sup>

## 1.4 In Living Organisms

Vroom and Yetton’s organisational style isn’t relevant to just teams of people. We can see it in the natural world too. Although *the majority* of cellular life on earth (by weight) is single celled organisms<sup>18</sup>, the existence of *humans* (to pick a single example) demonstrates that sometimes it’s better to try to mitigate Coordination Risk and work as a team, accepting the Complexity Risk and Communication Risk this entails. As soon as cells start working together, they either need to pass *resources* between them, or *control* and *feedback*.

<sup>16</sup>[https://en.wikipedia.org/wiki/Team\\_building](https://en.wikipedia.org/wiki/Team_building)

<sup>17</sup>[https://en.wikipedia.org/wiki/Gerald\\_Weinberg](https://en.wikipedia.org/wiki/Gerald_Weinberg)

<sup>18</sup>[http://www.stephenjagould.org/library/gould\\_bacteria.html](http://www.stephenjagould.org/library/gould_bacteria.html)

For example, in the human body, we have various systems<sup>19</sup>:

- The Respiratory System<sup>20</sup> which is responsible for ensuring that Red Blood Cells<sup>21</sup> are replenished with Oxygen, as well as disposing of Carbon Dioxide.
- The Digestive System<sup>22</sup> which is responsible for extracting nutrition from food and putting them in our Blood Plasma<sup>23</sup>.
- The Circulatory System<sup>24</sup> which is responsible for moving blood cells to all the rest of the body.
- The Nervous System<sup>25</sup> which is responsible for collecting information from all the parts of the body, dealing with it in the Brain<sup>26</sup> and issuing commands.
- The Motor System<sup>27</sup> which contains muscles and bones, and allows us to move about.

... and many others. Each of these systems contains organs, which contain tissues, which contain cells of different types. (Even cells are complex systems containing multiple different, communicating sub-systems.) There is huge Complexity Risk here: the entire organism fails if one of these systems fail (they are Single Points Of Failure<sup>28</sup>, although we can get by despite the failure of one lung or one leg say).

Some argue<sup>29</sup> that the human nervous system is the most complex known artifact in the universe: there is huge attendant Communication Risk to running the human body. But, given the success of humanity as a species, you must conclude that these steps on the evolutionary Risk Landscape have benefitted us in our ecological niche.

The key observation from looking at biology is this: most of the cells in the human body *don't get a vote*. Muscles in the motor system have

---

<sup>19</sup>[https://en.wikipedia.org/wiki/List\\_of\\_systems\\_of\\_the\\_human\\_body](https://en.wikipedia.org/wiki/List_of_systems_of_the_human_body)

<sup>20</sup>[https://en.wikipedia.org/wiki/Respiratory\\_system](https://en.wikipedia.org/wiki/Respiratory_system)

<sup>21</sup>[https://en.wikipedia.org/wiki/Red\\_blood\\_cell](https://en.wikipedia.org/wiki/Red_blood_cell)

<sup>22</sup>[https://en.wikipedia.org/wiki/Human\\_digestive\\_system](https://en.wikipedia.org/wiki/Human_digestive_system)

<sup>23</sup>[https://en.wikipedia.org/wiki/Blood\\_plasma](https://en.wikipedia.org/wiki/Blood_plasma)

<sup>24</sup>[https://en.wikipedia.org/wiki/Circulatory\\_system](https://en.wikipedia.org/wiki/Circulatory_system)

<sup>25</sup>[https://en.wikipedia.org/wiki/Nervous\\_system](https://en.wikipedia.org/wiki/Nervous_system)

<sup>26</sup><https://en.wikipedia.org/wiki/Brain>

<sup>27</sup>[https://en.wikipedia.org/wiki/Motor\\_system](https://en.wikipedia.org/wiki/Motor_system)

<sup>28</sup>[https://en.wikipedia.org/wiki/Single\\_point\\_of\\_failure](https://en.wikipedia.org/wiki/Single_point_of_failure)

<sup>29</sup><https://www.quora.com/What-is-the-most-complex-object-in-the-universe>



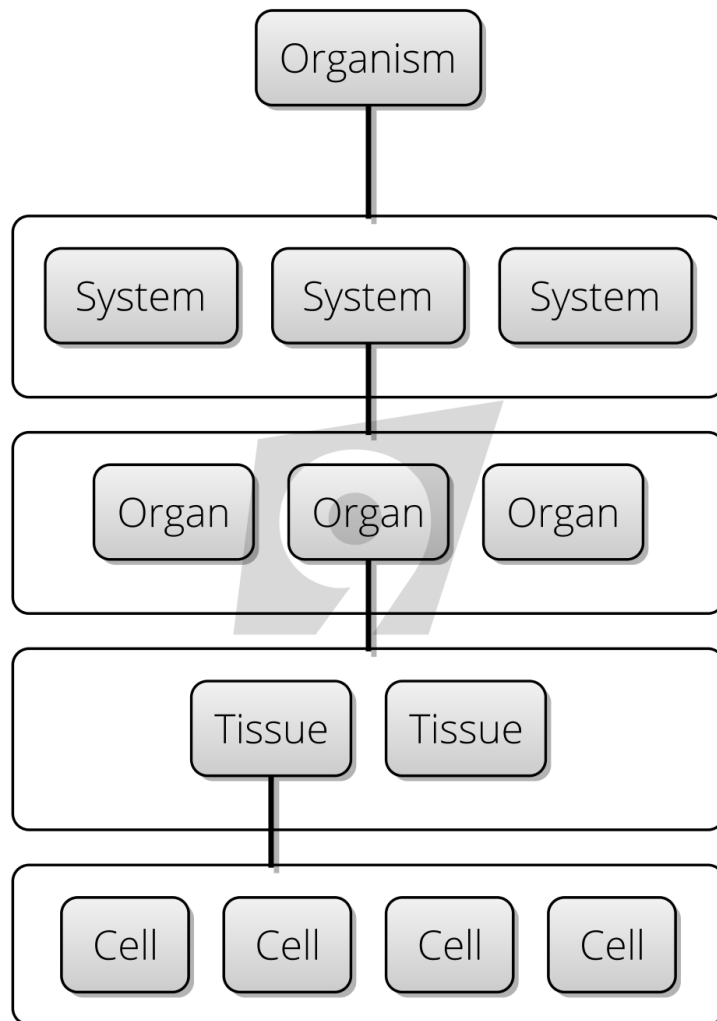


Figure 1.6: Hierarchy of Function in the Human Body

an **AI** or **AII** relationship with the brain - they do what they are told, but there are often nerves to report pain back. The only place where **CII** or **GII** *could* occur is in our brains, when we try to make a decision and weigh up the pros and cons.

This means that there is a deal: *most* of the cells in our body accede control of their destiny to “the system”. Living within the system of the human body is a better option than going it alone. Occasionally, due to mutation, we can end up with Cancer<sup>30</sup>, which is where one cell genetically “forgets” its purpose in the whole system and goes back to selfish individual self-replication (**UI**). We have White Blood Cells<sup>31</sup> in the body to shut down this kind of behaviour and try to kill the rogue cells. In the same way, society has a police force to stop undesirable behaviour amongst its citizens.

## 1.5 Large Organisations

Working in a large organisation often feels like being a cell in a larger organism. Just as cells live and die, but the organism goes on, in the same way, workers come and go from a large company but the organisation goes on. By working in an organisation, we give up self-control and competition and accept **AI** and **AII** power structures above us, but we trust that there is symbiotic value creation on both sides of the employment deal.

*Less* consultative decision making styles are more appropriate then when we don’t have the luxury of high-bandwidth channels for discussion, or when the number of parties rises above a room-full of people. As you can see from the table above, for **CII** and **GII** decision-making styles, the amount of communication increases non-linearly with the number of participants, so we need something simpler. As we saw in the Complexity Risk section, hierarchies are an excellent way of economizing on number of different communication channels, and we use these frequently when there are lots of parties to coordinate.

In large organisations, teams are created and leaders chosen for those teams precisely to mitigate Communication Risk. We’re all familiar with this: control of the team is ceded to the leader, who takes on

---

<sup>30</sup><https://en.wikipedia.org/wiki/Cancer>

<sup>31</sup>[https://en.wikipedia.org/wiki/White\\_blood\\_cell](https://en.wikipedia.org/wiki/White_blood_cell)

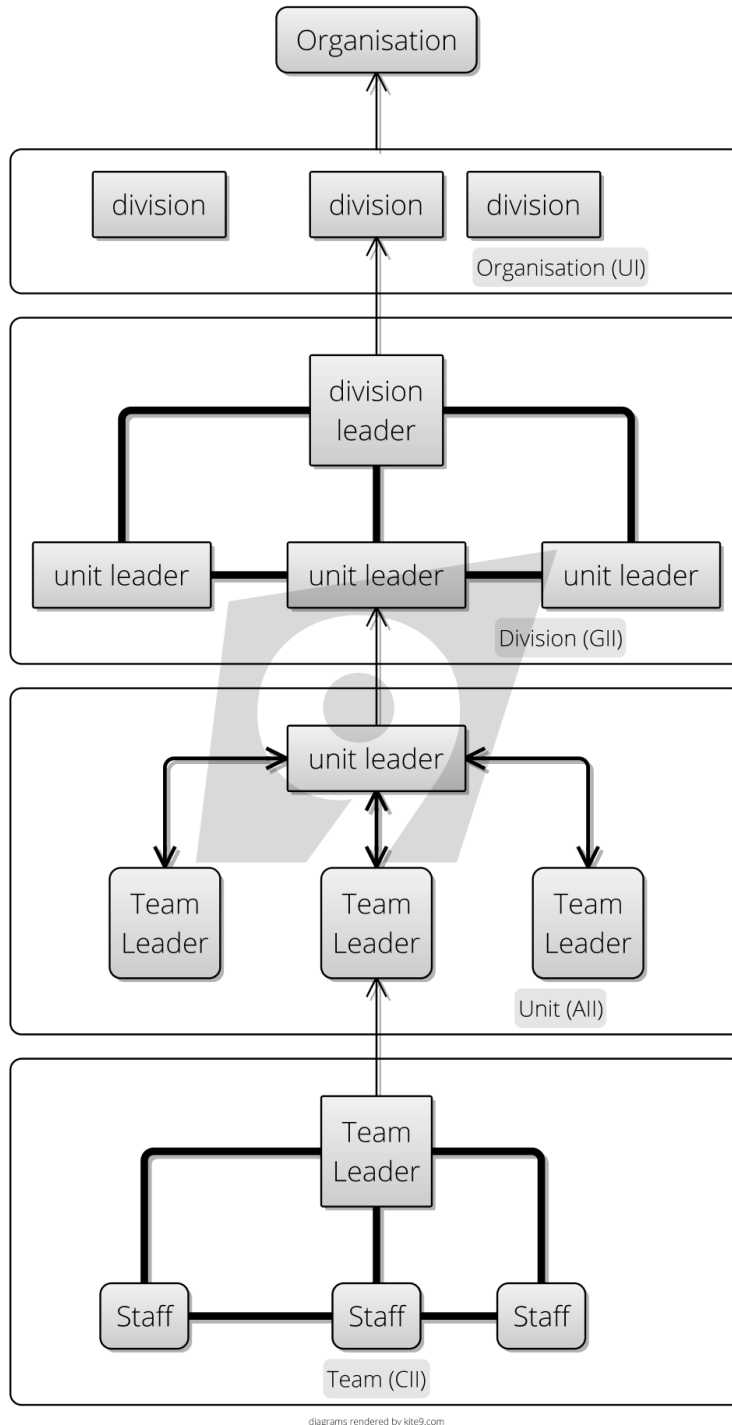


Figure 1.7: Hierarchy of Function in an Organisation

the role of ‘handing down’ direction from above, but also ‘reporting up’ issues that cannot be resolved within the team. In Vroom and Yetton’s model, this is moving from a **GII** or **CII** to an **AI** or **AII** style of leadership.

As shown in the diagram above, we end up with a hierarchy of groups, each having it’s own decision-making style. The team leader at the bottom level is a *decision maker* within his team, but moving up, doesn’t have decision making power in the next team up.. and so on.

Sometimes, parts of an organisation are encouraged *not* to coordinate, but to compete. In the diagram above, we have an M-Form<sup>32</sup> organisation, composed of *competing divisions*.

Clearly, this is just a *model*, it’s not set in stone and decision making styles usually change from day-to-day and decision to decision. The same is not true in our software - *rules are rules*.

## 1.6 In Software Processes

It should be pretty clear that we are applying the Scale Invariance rule to Coordination Risk: all of the problems we’ve described as affecting teams, also affect software, although the scale and terrain are different. Software processes have limited *agency* - in most cases they follow fixed rules set down by the programmers, rather than self-organising like people can (so far).

As before, in order to face Coordination Risk in software, we need multiple agents all working together. Coordination Risks (such as race conditions or deadlock) only really occurs where *more than one thing is happening at a time*. This means we are considering *at least* multi-threaded software and anything above that (multiple CPUs, servers, data-centres and so on).

### CAP Theorem

The CAP Theorem<sup>33</sup> has a lot to say about Coordination Risk. Imagine talking to a distributed database, where your request (*read* or *write*) can be handled by one of many agents.

---

<sup>32</sup>[https://en.wikipedia.org/wiki/Multi-divisional\\_form](https://en.wikipedia.org/wiki/Multi-divisional_form)

<sup>33</sup>[https://en.wikipedia.org/wiki/CAP\\_theorem](https://en.wikipedia.org/wiki/CAP_theorem)

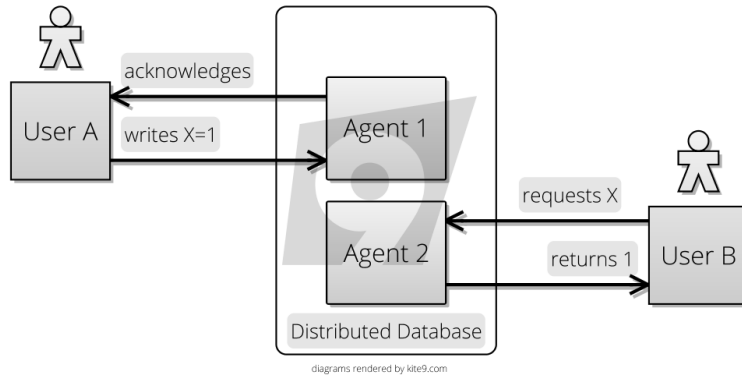


Figure 1.8: User A and User B are both using a distributed database, managed by Agents 1 and 2, whom each have their own Internal Model

In the diagram below, we have just two agents 1 and 2, in order to keep things simple. User A *writes something* to the database, then User B *reads it back* afterwards.

According to the CAP Theorem, there are three properties we could desire in such a system:

- **Consistency:** Every read receives the most recent value from the last write.
- **Availability:** Every request receives a response.
- **Partition tolerance:** The system can operate despite the isolation (lack of communication with) some of its agents.

The CAP Theorem states that this is a Trilemma<sup>34</sup>. That is, you can only have two out of the three properties.

There are plenty of resources on the internet that discuss this in depth, but let's just illustrate with some diagrams to show how this plays out. In our diagram example, we'll say that *any* agent can receive the read or write. So this might be a **GII** decision making system, because all the agents are going to need to coordinate to figure out what the right value is to return for a read, and what the last value written was. In these, the last write (setting X to 1) was sent to Agent 1 which then

<sup>34</sup><https://en.wikipedia.org/wiki/Trilemma>

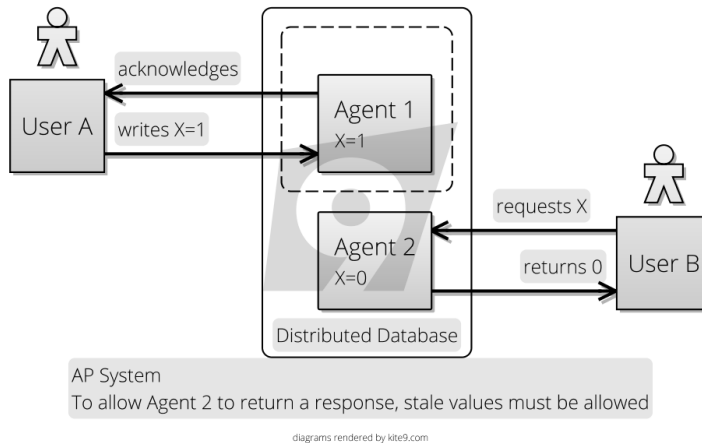


Figure 1.9: In an AP system, the User B will get back a *stale value* for X

becomes *isolated*, and can't be communicated with, due to network failure. What will User B get back?

## With an AP System

With AP, you can see that User B is getting back a *stale value*. AP scenarios lead to Race Conditions: Agent 1's availability determines what value User B gets back.

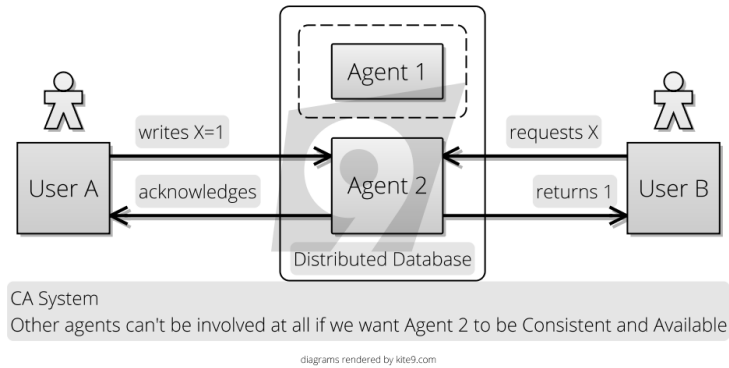
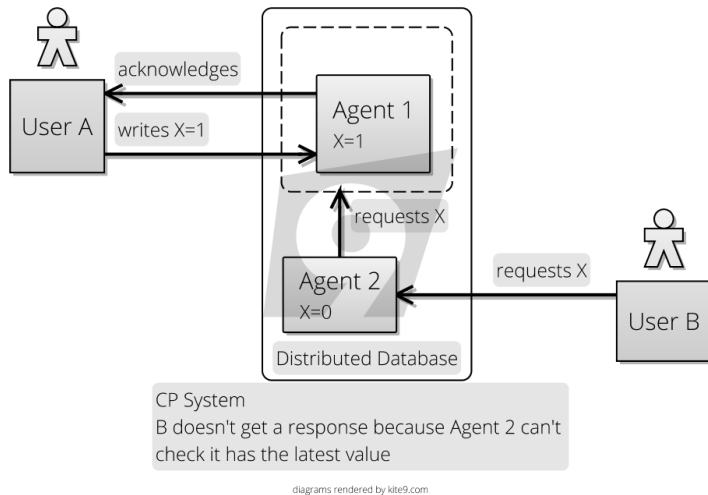


Figure 1.10: In an CA system, we can't have partition tolerance, so in order to be consistent a single Agent has to do all the work

### With an CP System



Where Agent 2 is left waiting for Agent 1 to re-appear, we are *blocked*. So CP systems lead to Deadlock scenarios.

### With an CA System

Finally, if we have a CA system, we are essentially saying that *only one agent is doing the work*. (You can't partition a single agent, after all). But this leads to Resource Allocation and **Contention** around use of

the scarce resource of Agent 2's attention. (Both Coordination Risk issues we met earlier.)

## Some Real-Life Examples

This sets an upper bound on Coordination Risk: we *can't* get rid of it completely in a software system, -or- a system on any other scale. Fundamentally, coordination problems are inescapable at some level. The best we can do is mitigate it by agreeing on protocols and doing lots of communication.

Let's look at some real-life examples of how this manifests in software.

### ZooKeeper

First, ZooKeeper<sup>35</sup> is an Open-Source datastore, which is used a lot for coordinating a distributed systems, and storing things like configuration information across them. If the configuration of a distributed system gets changed, it's important that *all of the agents in the system know about it*, otherwise... disaster.

This *seems* trivial, but it quickly gets out-of-hand: what happens if only some of the agents receive the new information? What happens if a datacentre gets disconnected while the update is happening? There are lots of edge-cases.

ZooKeeper handles this by communicating inter-agent with it's own protocol. It elects a **master agent** (via voting), turning it into an **AI**-style team. If the master is lost for some reason, a new leader is elected. *Writes* are then coordinated via the **master agent** who makes sure that a *majority of agents* have received and stored the configuration change before telling the user that the transaction is complete. Therefore, ZooKeeper is a CP system.

### Git

Second, git is a (mainly) write-only ledger of source changes. However, as we already discussed above, where different agents make incompatible changes, someone has to decide how to resolve the conflicts so that we have a single source of truth.

---

<sup>35</sup><https://zookeeper.apache.org>



The Coordination Risk just *doesn't go away*.

Since multiple users can make all the changes they like locally, and merge them later, Git is an AP system: individual users may have *wildly* different ideas about what the source looks like until the merge is complete.

## Bitcoin

Finally, Bitcoin (BTC)<sup>36</sup> is a write-only distributed ledger<sup>37</sup>, where agents *compete* to mine BTC, but also at the same time record transactions on the ledger. BTC is also AP, in a similar way to Git. But new changes can only be appended if you have the latest version of the ledger. If you append to an out-of-date ledger, your work will be lost.

Because it's based on outright competition, if someone beats you to completing a mining task, then your work is wasted. So, there is *huge* Coordination Risk.

For this reason, BTC agents coordinate into mining consortia<sup>38</sup>, so they can avoid working on the same tasks at the same time. But this in itself is a problem, because the whole *point* of BTC is that it's competitive, and no one entity has control. So, mining pools tend to stop growing before they reach 50% of the BTC network's processing power. Taking control would be politically disastrous<sup>39</sup> and confidence in the currency (such as there is) would likely be lost.

## 1.7 Communication Is For Coordination

So, now we have a fundamental limit on how much Coordination Risk we can mitigate. And, just as there are plenty of ways to mitigate Coordination Risk within teams of people, organisations or living organisms, so it's the case in software.

Earlier in this section, we questioned whether Coordination Risk was just another type of Communication Risk. However, it should be clear

---

<sup>36</sup><https://en.wikipedia.org/wiki/Bitcoin>

<sup>37</sup>[https://en.wikipedia.org/wiki/Distributed\\_ledger](https://en.wikipedia.org/wiki/Distributed_ledger)

<sup>38</sup>[https://en.bitcoin.it/wiki/Comparison\\_of\\_mining\\_pools](https://en.bitcoin.it/wiki/Comparison_of_mining_pools)

<sup>39</sup>[https://www.reddit.com/r/Bitcoin/comments/5fe9vz/in\\_the\\_last\\_24hrs\\_three\\_mining\\_pools\\_have\\_control/](https://www.reddit.com/r/Bitcoin/comments/5fe9vz/in_the_last_24hrs_three_mining_pools_have_control/)

after looking at the examples of competition, cellular life and Vroom and Yetton's Model that this is exactly *backwards*:

- Most single-celled life has no need for communication: it simply competes for the available resources. If it lacks anything it needs, it dies.
- There are *no* lines of communication on the **UI** decision-type. It's only when we want to avoid competition, by sharing resources and working towards common goals that we need to communicate.
- Therefore, the whole point of communication *is for coordination*.

In the next section, Map And Territory Risk, we're going to look at some new ways in which systems can fail, despite their attempts to coordinate.

# Map And Territory Risk

As we discussed in the section on Abstraction, our understanding of the world is entirely informed by the names we give things and the abstractions we create.

(In the same way, **Risk-First** is about *identifying patterns* within software development and calling them out.)

Our Internal Models are a model of the world based on these patterns, and their relationships.

So there is a translation going on here: observations about the arrangement of *atoms* in the world get turned into patterns of *information* (measured in bits and bytes).

Map And Territory Risk is the risk we face because we base our behaviour on our Internal Models rather than reality itself. It comes

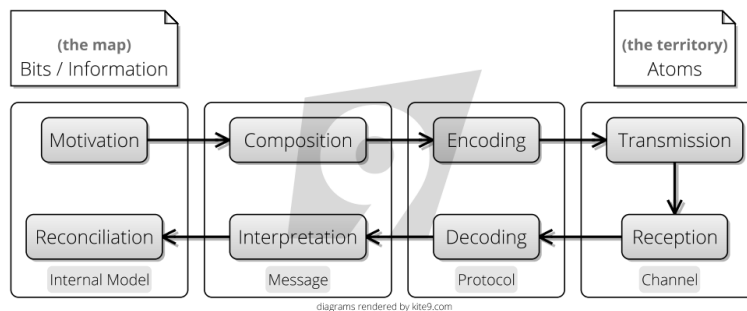


Figure 2.1: Maps and Territories, and Communication happening between them

from the expression “Confusing the Map for the Territory”, attributed to Alfred Korzybski:

“Polish-American scientist and philosopher Alfred Korzybski remarked that “the map is not the territory” and that “the word is not the thing”, encapsulating his view that an abstraction derived from something, or a reaction to it, is not the thing itself. Korzybski held that many people *do* confuse maps with territories, that is, confuse models of reality with reality itself.” - Map-Territory Relation, *Wikipedia*<sup>1</sup>

In this section, we’re going to make a case for analysing Map and Territory Risk along the same axes we introduced for Feature Risk, that is **Fitness**, **Audience** and **Evolution**. After that, we are going to widen the scope by looking at Map and Territory Risk within the context of **machines**, **people**, **hierarchies** and **markets**.

tbd - diagram of how our actions are based on the map, not the territory.

## 2.1 Fitness

In the picture shown here, from the Telegraph newspaper, the driver *trusted* the SatNav to such an extent that he didn’t pay attention to the road-signs around him, and ended up getting stuck.

This wasn’t borne of stupidity, but experience: SatNavs are pretty reliable. *So many times* the SatNav had been right, that the driver stopped *questioning its fallibility*.

So, there are two Map and Territory Risks here:

- The Internal Model of the *SatNav* contained information that was wrong: the track had been marked up as a road, rather than a path.
- The Internal Model of the *driver* was wrong: his abstraction of “the SatNav is always right” turned out to be only *mostly* accurate.

---

<sup>1</sup>[https://en.wikipedia.org/wiki/Mapterritory\\_relation](https://en.wikipedia.org/wiki/Mapterritory_relation)



A van got stuck on a narrow footpath when the driver took a wrong turn while blindly following his sat-nav. Photo: MEN

Figure 2.2: Sat Nav Blunder Sends Asda Van Crashing Narrow Footpath - Telegraph Newspaper

## 2.2 Internal Models as Dependencies, Features

What are the risks at play here? We've already looked in detail at the Dependency Risks involved in relying on something like a SatNav, in the Software Dependency Risk section. But here, we are really looking at the *Internal Models themselves* as a source of Dependency Risk too.

We could argue that the SatNav and the Driver's Internal Model had bugs in them. That is, they both suffer the Feature Implementation Risk we saw in the Feature Risk section. If a SatNav has too much of this, you'd end up not trusting it, and getting a new one. With your *personal* Internal Model, you can't buy a new one, but you may learn to *trust certain abstractions less*, as this driver did.

In the Feature Risk section, we broke down Feature Risk on three axes: **Fitness**, **Evolution** and **Audience**.

Lets do this again and see how each type of Feature Risk can manifest in the Internal Model:

As with Features in a product, Information in an internal model has at least these three dimensions:








Dimension	Feature Risk			Examples
Fit	 FI Fit Risk	 CI Conceptual Integrity Risk	 Im Implementation Risk	<ul style="list-style-type: none"> <li>• A filing cabinet containing too much junk.</li> <li>• Learning things that aren't useful.</li> <li>• Knowing how a car works, but actually needing to know how to drive.</li> <li>• Knowing how to program in one language, when another would be more appropriate.</li> <li>• Sat Nav had the wrong route.</li> <li>• Not quite remembering a recipe properly.</li> </ul>
	 FD Feature Design Risk	 R Regression Risk		<ul style="list-style-type: none"> <li>• Knowing outdated tools.</li> <li>• Writing last year's date on the cheque.</li> <li>• The bank sending letters to your old address.</li> <li>• Forgetting things</li> </ul>
	 FA Feature Access Risk	 Ma Market Access Risk		<ul style="list-style-type: none"> <li>• Memes.</li> <li>• Demand for courses.</li> <li>• Metrics.</li> <li>• Echo Chambers</li> <li>• Shared values which exclude certain people.</li> <li>• Ideas going "out of fashion".</li> </ul>
Audience				
Evolution				

Figure 2.3: Feature Risk, as manifested in the Internal Model

- **Fitness:** as discussed above with the SatNav example, this is how closely the information matches reality, and how *useful that is to us* (models that contain too much detail are as bad as models with too little).
- **Audience:** is all about how a piece of information is *shared* between many Internal Models, and it's this we are going to address further now.
- **Evolution:** is all about how Internal Models change when they meet reality, and we'll cover that last.

## 2.3 Audience

We already know a lot about Internal Models and audience, as these have been the subject of previous sections:

- We know from looking at Communication Risk that communication allows us to *share* information between Internal Models.
- We know from Coordination Risk the difficulties inherent in aligning Internal Models so that they cooperate.
- Job markets show us that there is demand for people with certain *skills*. This demonstrates to us that Market Risk is as applicable to Internal Models containing certain information as it is to products containing Features. This was the focus of the Ecosystem discussion in Boundary Risk.

... And, we're all familiar with *memes*:

"A meme acts as a unit for carrying cultural ideas, symbols, or practices, that can be transmitted from one mind to another through writing, speech, gestures, rituals, or other imitable phenomena with a mimicked theme." - Meme, *Wikipedia*<sup>2</sup>

Therefore, we should be able to track the rise-and-fall of *ideas* much as we can track stock prices. And in effect, this is what Google Trends<sup>3</sup>

---

<sup>2</sup><https://en.wikipedia.org/wiki/Meme>

<sup>3</sup><https://trends.google.com>

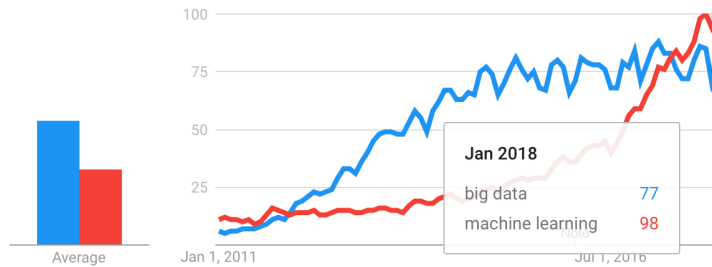


Figure 2.4: Relative popularity of “Machine Learning” and “Big Data” as search terms on Google Trends, 2011-2018

does. In the chart below, we can see the relative popularity of two search terms over time. This is probably as good an indicator as any of the audience for an abstraction at any point in time.

## Example: Hype Cycles

Most ideas (and most products) have a slow, hard climb to wide-scale adoption. But some ideas seem to disperse much more rapidly and are picked up quickly because they are exciting and promising, having greater “memetic potential” within society. One way this evolution manifests itself in the world is through the Hype Cycle<sup>4</sup>:

“The hype cycle is a branded graphical presentation developed and used by the American research, advisory and information technology firm Gartner, for representing the maturity, adoption and social application of specific technologies. The hype cycle provides a graphical and conceptual presentation of the maturity of emerging technologies through five phases.” - Hype Cycle, *Wikipedia*

The five phases (and the “Hype” itself) are shown in the chart below, with the thick black line being “Hype”:

<sup>4</sup>[https://en.wikipedia.org/wiki/Hype\\_cycle](https://en.wikipedia.org/wiki/Hype_cycle)



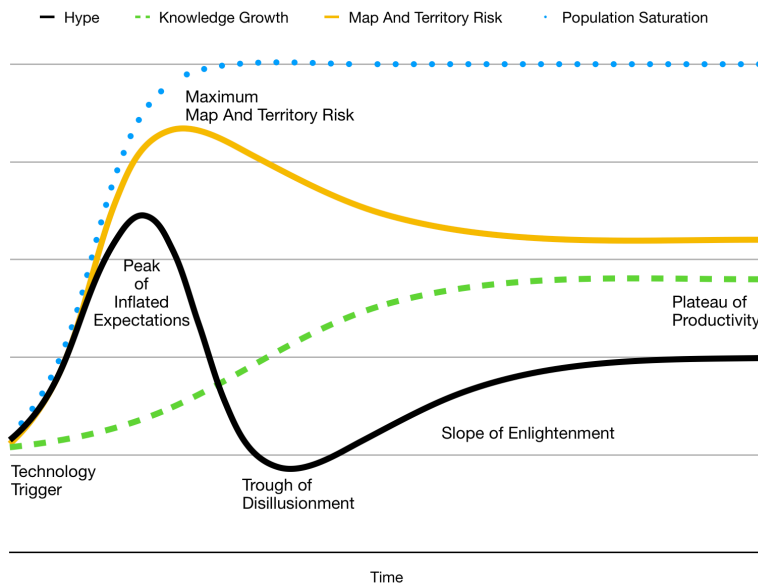


Figure 2.5: Hype Cycle, along with Map & Territory Risk

Also in this diagram we are showing where the hype originates:

- The **saturation** of the idea within the audience (a dotted line).
- The **amount known** about the idea by the audience (a Learning Curve, if you will, a dashed line).

Both of these are modelled with Cumulative Distribution<sup>5</sup> curves. From these two things, we can figure out where our maximum Map and Territory Risk lies: it's the point where awareness of an idea is furthest from the understanding of it. This acts as a "brake" on the **hype** around the idea, corresponding to the "Trough of Disillusionment".

Where the **saturation** and **knowledge** grow together, there is no spike in Map and Territory Risk and we don't see the corresponding "Trough of Disillusionment" at all, as shown in this chart:

<sup>5</sup>[https://en.wikipedia.org/wiki/Cumulative\\_distribution\\_function#Use\\_in\\_statistical\\_analysis](https://en.wikipedia.org/wiki/Cumulative_distribution_function#Use_in_statistical_analysis)

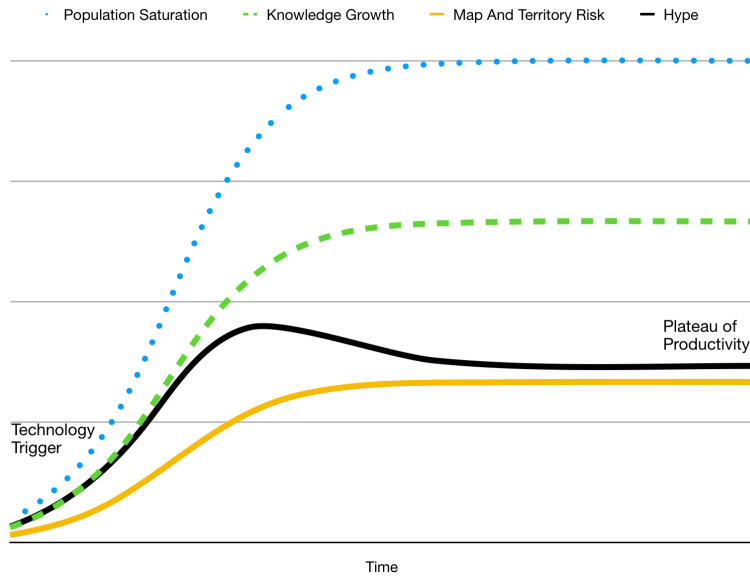


Figure 2.6: Hype Cycle 2: Slower growth of Map and Territory Risk means no “Trough of Disillusionment”

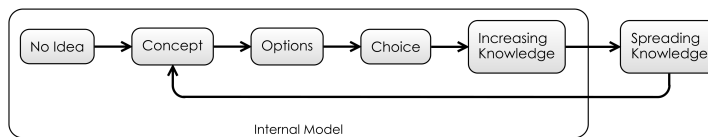


Figure 2.7: Spread of information between Internal Models

## 2.4 Evolution

The section on Communication Risk introduced the following model for ideas:

But what happens next? As we saw in Boundary Risk, the **Peter Principle** applies, people will use dependencies up to the point when they start breaking down.

## Example: Metrics

Let's dive into a specific example now: someone finds a useful new metric that helps in evaluating performance.

It might be:

- **SLOC (Source Lines Of Code):** i.e. the number of lines of code each developer writes per day/week whatever.
- **Function Points:** the number of function points a person on the team completes, each sprint.
- **Code Coverage:** the number of lines of code exercised by unit tests.
- **Response Time:** the time it takes to respond to an emergency call, say, or to go from a feature request to production.
- **Release cadence:** number of releases a team performs, per month, say.

With some skill, they may be able to *correlate* this metric against some other more abstract measure of success. For example:

“quality is correlated with more releases” “user-satisfaction is correlated with SLOC” “revenue is correlated with response time”

Because the *thing on the right* is easier to measure than *the thing on the left*, it becomes used as a proxy (or, Map) for the thing they are really interested in (the Territory). At this point, it's *easy* to communicate this idea with the rest of the team, and *the market value of the idea is high*: it is a useful representation of reality, which is shown to be accurate at a particular point in time.

But *correlation* doesn't imply *causation*. The *cause* might be different:

- quality and number of releases might both be down to the simplicity of the product.
- user satisfaction and SLOC might both be down to the calibre of the developers.
- response time and revenue might both be down to clever team planning.

Metrics are seductive because they simplify reality and are easily communicated. But they *inherently* contain Map and Territory Risk: By relying *only* on the metrics, you're not really *seeing* the reality.

The devil is in the detail.

## Reality Evolves

In the case of metrics, this is where they start being used for more than just indicators, but as measures of performance or targets:

- If a team is *told* to do lots of releases, they will perform lots of releases *at the expense of something else*.
- If team members are promoted according to SLOC, they will make sure their code takes up as many lines as possible.
- In the UK, ambulances were asked to wait before admitting patients to Emergency wards, in order that hospitals could meet their targets<sup>6</sup>.

Some of this seems obvious: *Of course* SLOC is a terrible measure performance! We're not that stupid anymore. The problem is, it's not so much the *choice* of metric, but the fact that *all* metrics merely approximate reality with a few numbers. The map is *always* simpler than the territory, therefore there can be no perfect metrics.

In the same way that markets evolve to demand more features, our behaviour evolves to incorporate new ideas. The more popular an idea is, the more people will modify their behaviour as a result of it, and the more the world will change. Will the idea still be useful as the world adapts? Although the Hype Cycle model doesn't cover it, ideas and products all eventually have their day and decline in usefulness.

## Bad Ideas

There are plenty of ideas which *seem a really good idea at the time* but then end up being terrible. It's only as we *learn about the products* and realize the hidden Map and Territory Risk that we stop using them.

---

<sup>6</sup>[https://en.wikipedia.org/wiki/NHS\\_targets](https://en.wikipedia.org/wiki/NHS_targets)

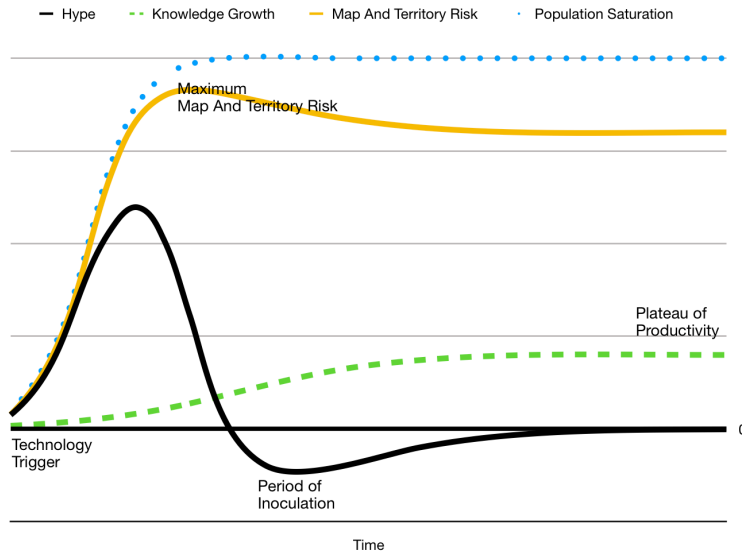


Figure 2.8: Hype Cycle For Something that turns out to be a *bad* idea

While SLOC is a minor offender, CFCs<sup>7</sup> or Leaded Petrol<sup>8</sup> are more significant examples.

The following Hyph Cycle chart shows an initially promising idea that turns out to be terrible, and there is a “Period of Inoculation” where the population realise their mistake. There is “negative hype” as they work to phase out the offending idea:

## 2.5 Humans and Machines

In the example of the SatNav, we saw how the *quality* of Map and Territory Risk is different for *people* and *machines*. Whereas people *should* be expected show skepticism to new (unlikely) information, our databases accept it unquestioningly. *Forgetting* is an everyday, usually benign part of our human Internal Model, but for software systems it is a production crisis involving 3am calls and backups.

<sup>7</sup><https://en.wikipedia.org/wiki/Chlorofluorocarbon>

<sup>8</sup><https://en.wikipedia.org/wiki/Tetraethyllead>

For Humans, Map and Territory Risk is exacerbated by cognitive biases<sup>9</sup>:

“Cognitive biases are systematic patterns of deviation from norm or rationality in judgment, and are often studied in psychology and behavioral economics.” - Cognitive Bias, *Wikipedia*

There are *lots* of cognitive biases. But let’s just look at a couple that are relevant to Map and Territory Risk:

- **Availability Heuristic:** People overestimate the importance of knowledge they have been exposed to.
- **The Ostrich Effect:** Which is where dangerous information is ignored or avoided because of the emotions it will evoke.
- **Bandwagon Effect:** People like to believe things that other people believe. Could this be a factor in the existence of the Hype Cycle?

## 2.6 Hierarchical Organisations

Map And Territory Risk “trickles down” through an organisation. The higher levels have an outsize ability to pervert the incentives at lower levels because once an organisation begins to pursue a “bullshit objective”, the whole company can align to this.

The Huffington Post<sup>10</sup> paints a brilliant picture of how Volkswagen managed to get caught faking their emissions tests. As they point out:

“The leadership culture of VW probably amplified the problem by disconnecting itself from the values and trajectory of society, by entrenching in what another executive in the auto industry once called a “bullshit-castle”. . . No engineer wakes up in the morning and thinks: OK, today I want to build devices that deceive our customers and destroy our planet. Yet it happened. Why? Because of hubris at the top.” - Otto Scharmer, *Huffington Post*.

---

<sup>9</sup>[https://en.wikipedia.org/wiki/List\\_of\\_cognitive\\_biases](https://en.wikipedia.org/wiki/List_of_cognitive_biases)

<sup>10</sup>[https://www.huffingtonpost.com/otto-scharmer/the-fish-rots-from-the-he\\_b\\_8208652.html](https://www.huffingtonpost.com/otto-scharmer/the-fish-rots-from-the-he_b_8208652.html)

This article identifies the following process:

- **De-sensing:** VW Executives ignored *The Territory* society around them (such as the green movement), ensuring their maps were out of date. The top-down culture made it hard for reality to propagate back up the hierarchy.
- **Hubris/Absencing:** They pursued their own metrics of *volume* and *cost*, rather than seeking out others (a la the Availability Heuristic Bias). That is, focusing on their own *Map*, which is *easier* than checking the *Territory*. (See Hubris in the Agency Risk section).
- **Deception:** Backed into a corner, engineers had no choice but to find “creative” ways to meet the metrics.
- **Destruction:** Eventually, the truth comes out, to the detriment of the company, the environment and the shareholders. As the article’s title summarizes “A fish rots from the head down”.

## Personal Example

A similar (but less catastrophic) personal story from a bank I worked at, where the objectives end up being mis-aligned *within the company*:

1. My team had been tasked with building automated “smoke tests” of an application. But this was bullshit: We only needed to build these *at all* because the application was so complex. The reason it was so complex was. . .
2. The application was being designed within a “Framework” constructed by the department. However, the framework was only being used by this one application. Building a “reusable” framework which is only used by a single application is bullshit. But, we had to do this because. . .
3. The organisational structure was created along a “matrix”, with “business function” on one axis and “functional area” on another. Although we were only building the application for a single business function, it was expected to cater with all the requirements from the an entire “functional area”. This was bullshit too, because. . .

4. The matrix structure was largely the legacy of a recent merger with another department. As Conway's Law predicts, our software therefore had to reflect this structure. But this was bullshit because. . .
5. The matrix structure didn't represent reality in any useful way. It was designed to pacify the budget committee at the higher level, and try to demonstrate attributes such as *control* and *governance*. But this was bullshit too, because. . .
6. The budget that was given to our department was really based on how much fear the budget holders currently had of the market regulators. But this was bullshit too, because. . .
7. At a higher level, the executives had realised that our division wasn't one of the bank's strategic strengths, and was working to close it all down anyway.

When faced with so many mis-aligned objectives, it seemed completely hopeless to concentrate on the task at hand. But then, a colleague was able to nihilistically add one final layer to this onion by saying:

8. "It's all about chasing money, which is bullshit, because life is bullshit."

## Picking Fights

It feels like there's no way back from that.

All of life might well be a big Map and Territory illusion. But let's analyse just a bit:

- At each layer, the objectives changed. But, they impacted on the objectives of the layer below.
- Therefore, it seems like the more layers you have, the less likely it is that your objectives become inconsistent between the lower and higher levels.
- On a new project, it seems like a good idea to model this stuff: does the objective of the work you're about to undertake "align" with the objectives at a higher level?



Trying to spot Map and Territory Risk ahead-of-time in this manner seems like a useful way of trying to avoid Vanity Projects, and, if you are good at it, allows you to see which Goals in the organisation are fragile and likely to change. However, usually, if you are working in a team, you have limited agency to decide which projects you feel are valuable.

This comes down to a personal decision: do you want to spend time working on projects that you know are going in the bin? Some developers have the attitude that, so long as they get paid, it doesn't matter. But others are in it for the satisfaction of the work itself, so this ends up being a personal call.

(This theme will be developed further in Staging and Classifying.)

## 2.7 Markets

So far, we've considered what happens to individuals, teams and organisations when told to optimise around a particular objective. In Coordination Risk we looked at how Communication was critical for Coordination to happen. And, as we've already discussed, Abstraction is a key part of communication.

The languages we adopt or create are *sets of useful abstractions* that allow us to communicate. But what happens when this goes wrong?

Inadequate Equilibria<sup>11</sup> by Eleizer Yudkovsky, looks at how perverse incentive mechanisms break not just departments, but entire societal systems. He highlights one example involving *academics* and *grant-makers* in academia:

- It's not very apparent which scientists are better than which other scientists.
- One proxy is what they've published (scientific papers) and where they've published (journals).
- Universities want to attract research grants, and the best way to do this is to have the best scientists.
- Because "best" isn't measureable, they use the proxy.

---

<sup>11</sup><https://equilibriabook.com>

- Therefore, immense power rests in the hands of the journals, since they can control the money-proxy.
- Therefore, journals are able to charge large amounts of money to universities for subscriptions.

“Now consider the system of scientific journals. . . Some journals are prestigious. So university hiring committees pay the most attention to publications in that journal. So people with the best, most interesting-looking publications try to send them to that journal. So if a university hiring committee paid an equal amount of attention to publications in lower-prestige journals, they’d end up granting tenure to less prestigious people. Thus, the whole system is a stable equilibrium that nobody can unilaterally defy except at cost to themselves.” - *Inadequate Equilibria*, *Eliezer Yudkovsky*<sup>12</sup>

As the book points out, while everyone *persists* in using an inadequate abstraction, the system is broken. However, Coordination would be required for everyone to *stop* doing it this way, which is hard work. (At least within a hierarchy, Maps can get fixed.)

This is a *small example* from a much larger, closely argued book, and it’s worth taking the time to read a couple of the chapters on this interesting topic.

As usual, this section forms a grab-bag of examples in a complex topic. But it’s time to move on as there is one last stop we have to make on the Risk Landscape, and that is to look at Operational Risk.

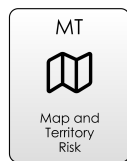
(NB: The Hype Cycle model is available in **Numbers** form here<sup>13</sup>.)

(talk about how operational risk is an extension of this). tbd

---

<sup>12</sup><https://equilibriabook.com/molochs-toolbox/>

<sup>13</sup><https://github.com/risk-first/website/blob/master/RiskMatrix.numbers>



- Risks due to the differences between reality and the internal model of reality, and the assumption that they are equivalent.

Figure 2.9: Map And Territory Risk



# **Part III**

## **Preview**



book1/Part3.md practices/Estimates.md





# Glossary

## Abstraction

## Feedback Loop

## Goal In Mind

## Internal Model

The most common use for Internal Model is to refer to the model of reality that you or I carry around in our heads. You can regard the concept of Internal Model as being what you *know* and what you *think* about a certain situation.

Obviously, because we've all had different experiences, and our brains are wired up differently, everyone will have a different Internal Model of reality.

Alternatively, we can use the term Internal Model to consider other viewpoints: - Within an organisation, we might consider the Internal Model of a *team of people* to be the shared knowledge, values and working practices of that team. - Within a software system, we might consider the Internal Model of a single processor, and what knowledge it has of the world. - A codebase is a team's Internal Model written down and encoded as software.

An internal model *represents* reality: reality is made of atoms, whereas the internal model is information.

**Meet Reality**

**Risk**

**Attendant Risk**

**Hidden Risk**

**Mitigated Risk**

**Take Action**