Evolutionary Architecture: What can we learn from Nature?

Chris Howe-Jones

4th April 2019

Who am I?

Name: `Chris Howe-Jones` **[Technical Navigator]**
Twitter: `@agile_geek`
Github: `https://github.com/chrishowejones`

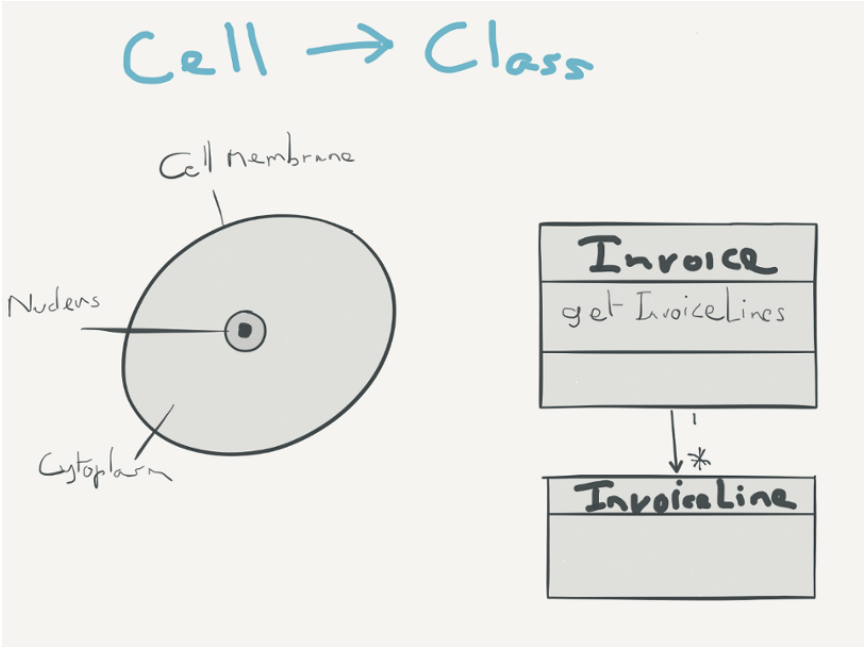
- Been programming for over 40 years, professionally for over 30.
- I've been designing software systems for over 25 years
- Started in monolithic homogeneous environment - mainframes
- Automating basic processes in a large government org that changed rules every couple of years with change of the minister but within parameters mainly controlled by the speed with which the org could respond.

- Environment was constrained and could only change at a fairly slow rate
- In the 90's/00's software centred companies competed with new features as their selling point so speed of change became a competitive advantage.
- Now we work in markets driven by customers that interact directly rather than indirectly.
- Companies are built up around providing services that don't exist before the company introduces it.
  - Facebook (although arguably MySpace was also in that field)
- Companies that are build a business model around a uniqueness - Google
- Companies even more frequently provide solutions to problems that already have solutions but they add some additional novelty, usually in the form of features that provide a different form of interaction with the customer that didn't exist before the software was written.
  - Apple does this but often in the form of physical design combined with software e.g. the iPod wheel as the hook and iTunes which provided an interface that other MP3 download sites didn't have.
  - Slack does what IRC has done for 30 years but provided a UI that was more tactile and visual (introducing previews for images, emoji) but also, the killer feature, was a persistent history making the communication more async as you don't need to be online at the same time. Also providing a fermium model and removing the hosting barriers of some competitors. Having the message limits meant you can be away from the convo for a while but not too long introducing a compulsion to check regularly. Also focusing on teams rather than threaded comments, initially they implemented everything as a continuous stream which encouraged constant checking which is encourages compulsive behaviour.
- We see companies starting with one vision of their offering but changing direction, sometimes quite radically.
  - Stuart Butterfield has done this a few times:

* Ludicorp founded 2002 to build a game (called Game Neverending)
* 2004 - game failed to gain traction but embedded photo-uploading was popular.
* 2005 - game crumbled but Flickr rose from ashes
* 2006 - Flickr sold to Yahoo
* 2008 - Stuart left Yahoo to found Tiny Spec creating a game called Glitch.
* 2010 - raised Series A funding, no detail about game but it was described as web-based massively-multiplayer game (Massively Mulitplayer Online Role-Playing Game MMORPG) epected launch 2010.
* 2011 - raised more funding. Invite-based Beta. Launch but returned to Beta to make it easier for new players
* 2012 - became obvious Glitch could not attract audience large enough to sustain itself. Glitch dead- Tiny Spec was not.
* 2013 - internal comms tool used to share ideas between the US and Canadian offices of Glitch recognised as valuable
* 2014 - Slack was announced.

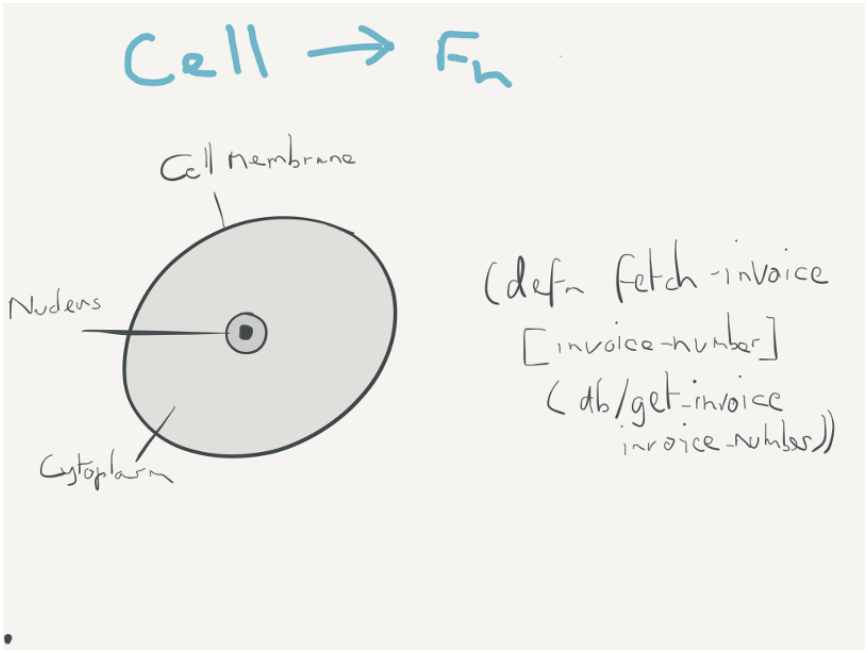
- Seeking different analogies - engineering is not a great one are there other perspectives?
  - Civil Engineering - Building the Tyne Bridge. No one widens the Tyne by 300 miles and relocates it to the South Pole mid project.
  - Engineering in Aerospace - flying a plane or guiding a spacecraft is defined by simple Newtonian physics. Lots of variables but simple equations.
  - What about biology?

Mapping Software to Biology



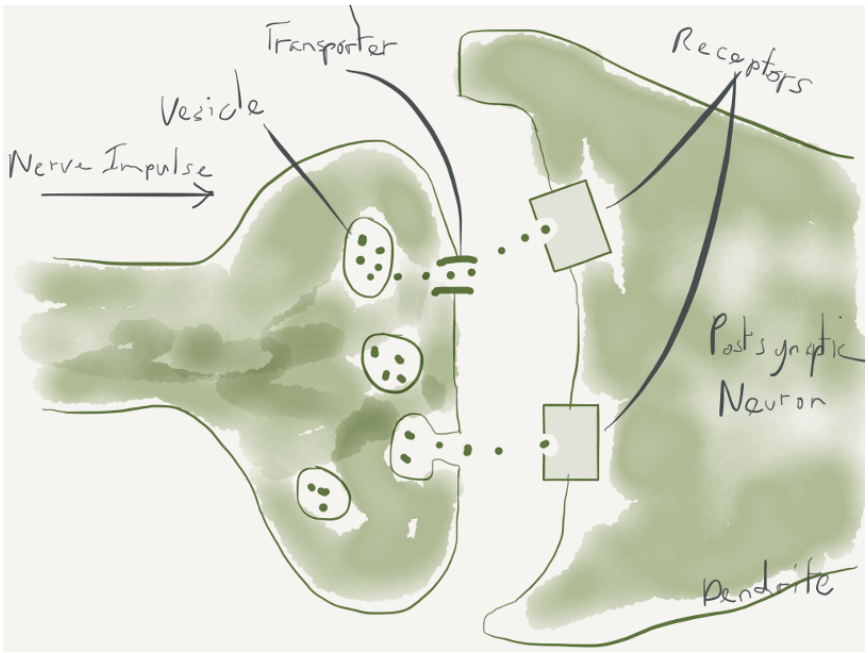
- How would software design map to biological systems?
- How do classes/functions map to cells?
- Cells are self contained - but communication or cooperate to provide higher functions

Mapping Software to Biology



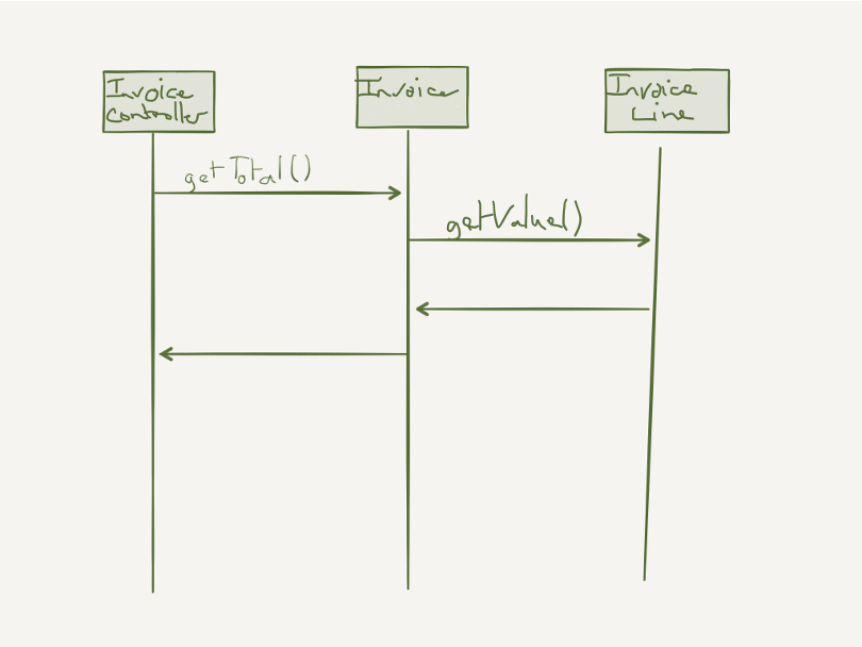
- Message passing like neurochemical transmission
  - different types:
    - neurotransmitters - message calling between methods/functions? (chemical synapse - such as gultamate)
    - blood - async message based systems, stream processing
    - nerve endings/hormones - endpoints recieving stimuli in the form of 'data' from the outside world.

Messages



- Message passing like neurochemical transmission
  - different types:
    - neurotransmitters - message calling between methods/functions?
    - blood - async message based systems, stream processing
    - nerve endings/hormones - endpoints recieving stimuli in the form of 'data' from the outside world.

Messages

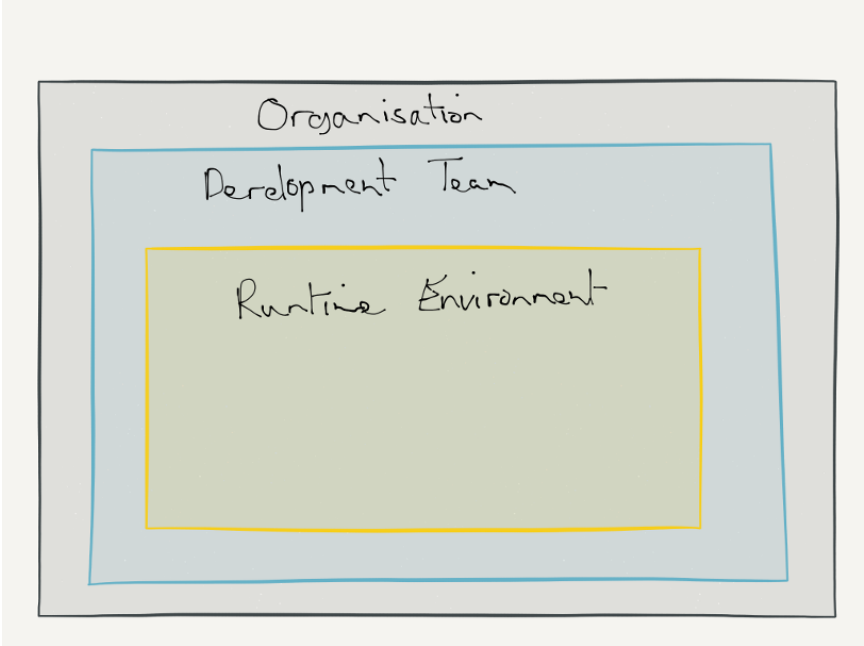


- Message passing like neurochemical transmission
  - different types:
    - neurotransmitters - message calling between methods/functions?
    - blood - async message based systems, stream processing
    - nerve endings/hormones - endpoints recieving stimuli in the form of 'data' from the outside world.

S/w Environments -> Ecosystems

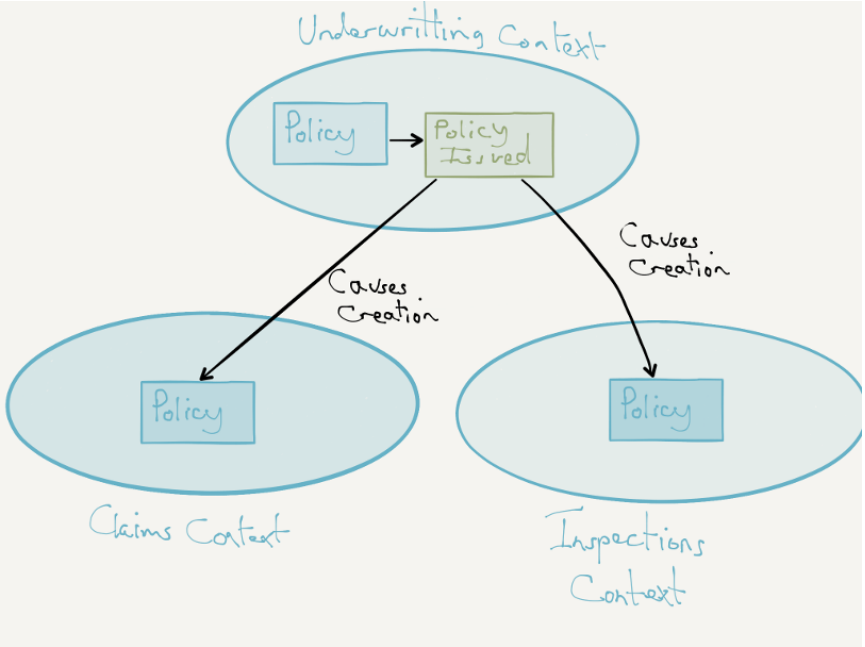
- Ecosystem in nature
- At various levels
- Complex system subject to changes in temp, moisture, destructive forces.

S/w Environments -> Ecosystems



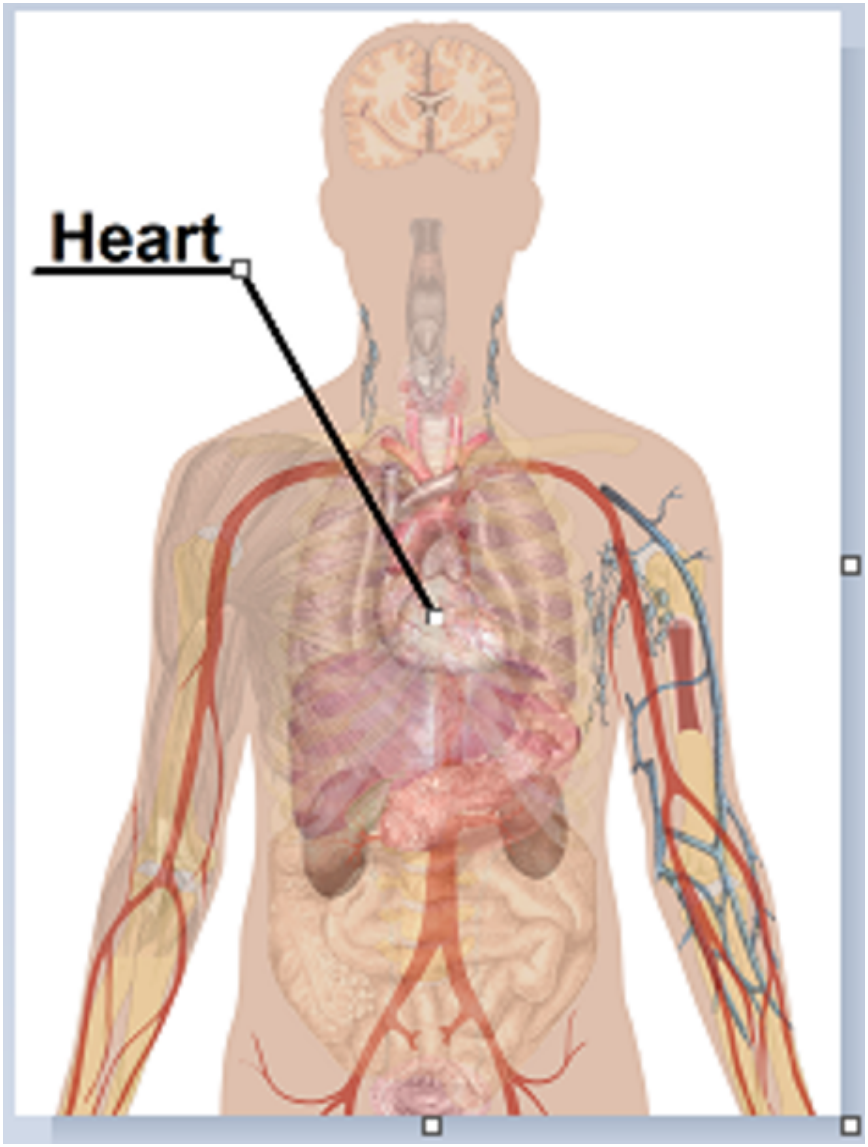
- Ecosystem at various levels
  - One of the issues is what state is the Ecosystem in?
  - This is not obvious, how do we measure or detect what the Ecosystem is doing as a whole while it's active?
  - How are parts of the ecosystem reacting in coordination to others?
- Changes to the Ecosystem are changes to features, requirements, etc. Large changes are changes in climate, different ecosystem, etc.
- Moving a specialised organism to a completely different ecosystem means it will struggle to survive or may simply die immediately.
- Evolution is the organism changing over time to better exploit its ecosystem - the mechanism of evolution is the development team.

DDD Bounded Contexts



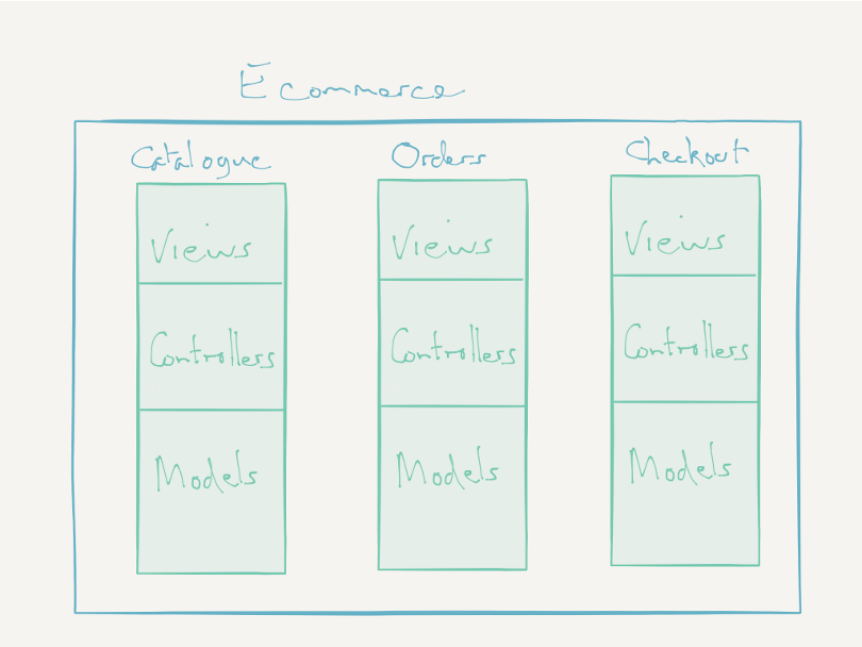
- Are DDD bounded contexts the equivalent of separate organisms?
- They evolve at there own rates. They may be affected by the same stimuli but they don't share internal structures.
- Abstractions within software (interfaces and data structures) are analogous to membranes and sensory mechanisms in organisms.

Monoliths



Monoliths are like a single organism and the 'bounded context' are the organs . Organs can share nutrients, communication mechanisms (ganglia, neurotransmitters).

Monoliths

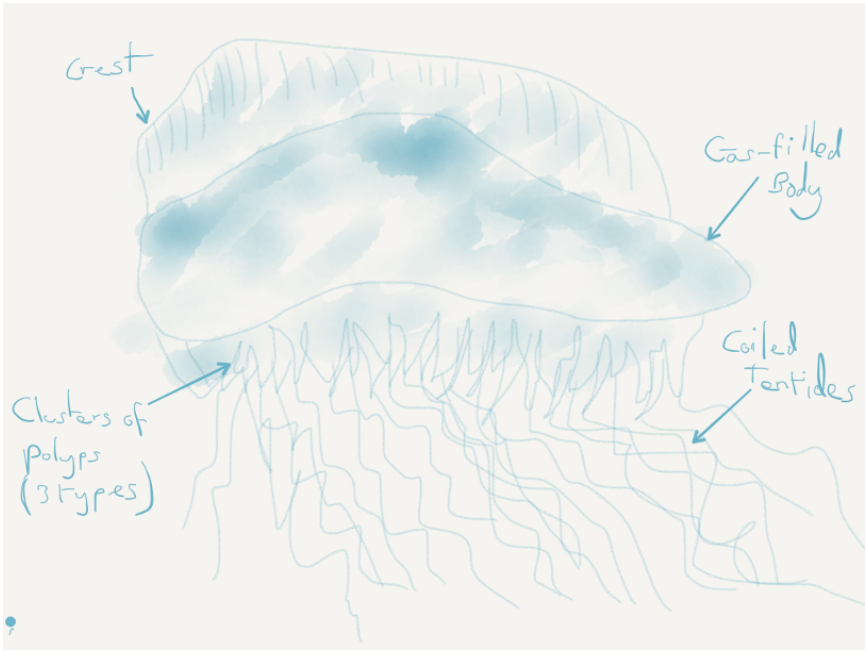


Packages or namespaces are bounded context equivalent to organs Comms via in process calls

Services

Services are individual organisms the boundaries of which are the 'bounded context'. Each service is separate. They evolve at their own rates. They may be affected by the same stimuli but they don't share internal structures.

Microservices

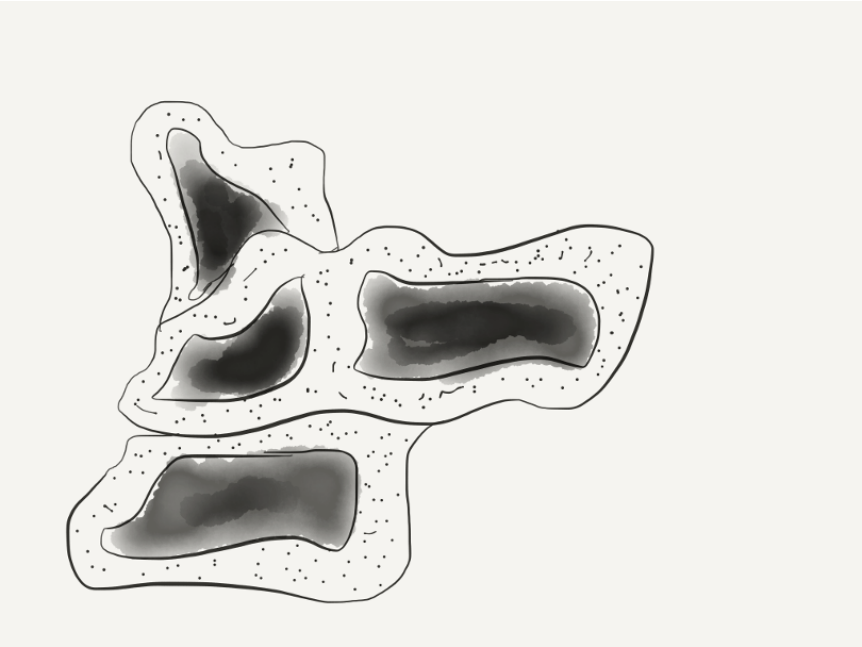


Microservices are like a Siphonophorae, like a Portuguese man o' war. A bounded context is a colonial organism made up of small organisms that cooperate to get a task done. There are different types of organism with specialised functions zooids or polyps - dactylozooid (defense), gonozooid (reproduction), and gastrozooid (feeding)

Problems

Negative Environmental Factors -> Design Faults

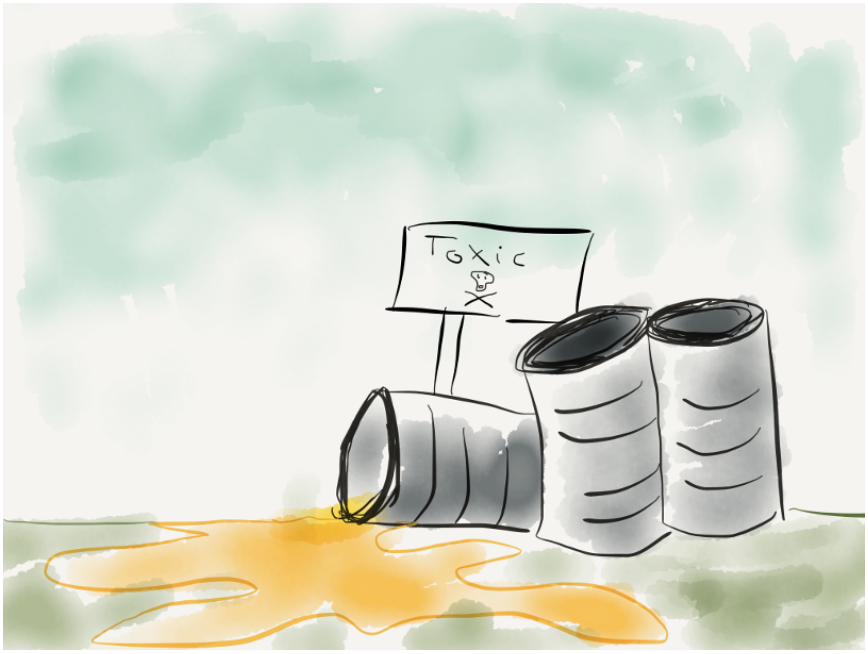
Problems



Poor architecture in:

- Monoliths - cancer(uncontrolled growth), disease (breakdown of shared classes, namespaces, etc).
- Stress factors in cells (hormones) cause DNA knots that interfere with RNA transcription.

Problems



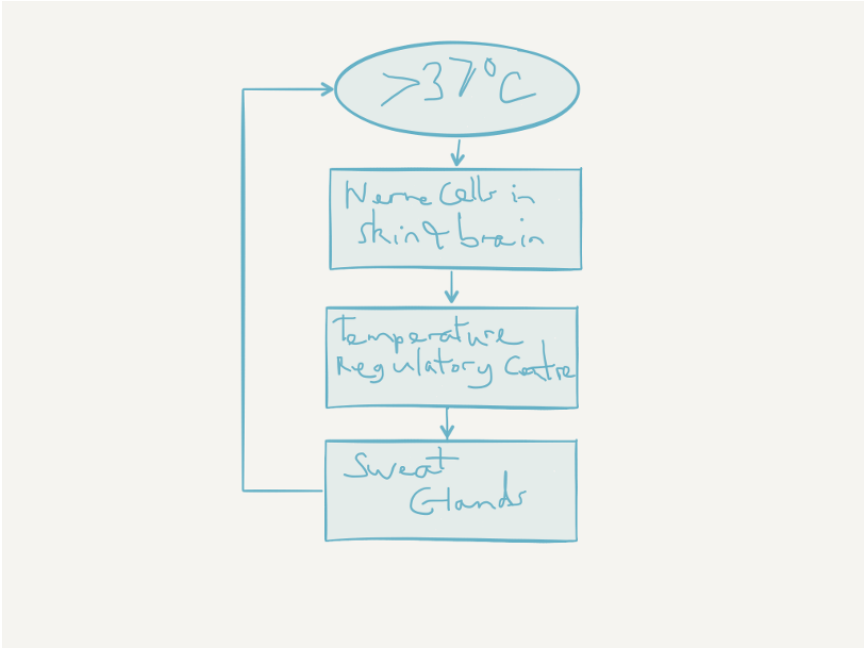
Poor architecture in:

- Services - environmental changes - polution, poisonous environmental factors, extremes of heat and cold. Each individual service is subject to disease, cancer, etc.
- Microservices - less impacted by disease, cancer as they're smaller/simpler organisms less to go wrong. Environmental changes in the colony effect the whole colony. We can lose an individual in the organism type in the colony and continue.

Individual Organisms mechanisms

- So if each organism is a monolith, a service or a microservice that operates within the bounds of it's ecosystem.
- How do they sense and communicate with the ecosystem or other 'organisms'?

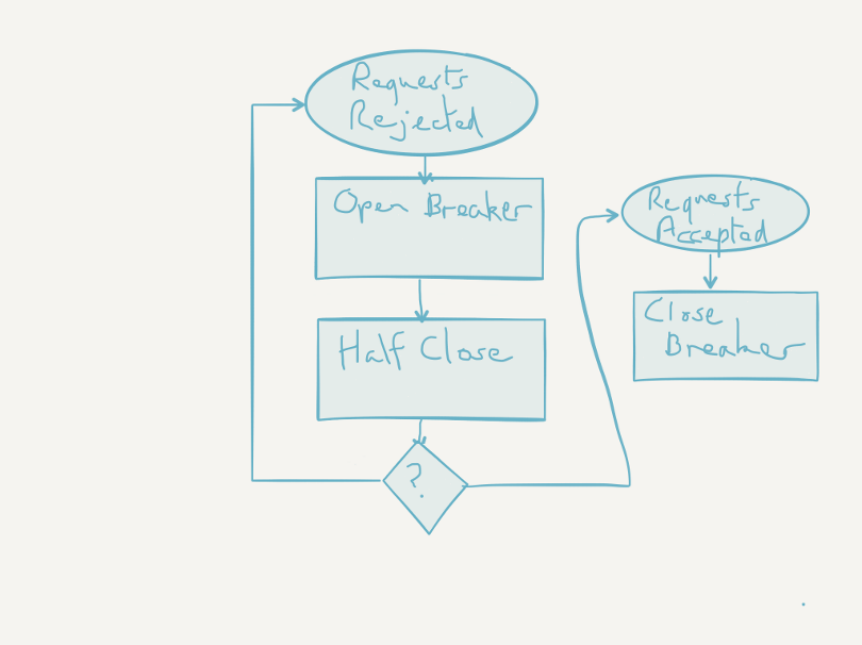
-ve feedback mechanisms



- Damp down the effect of a stimulus
- Homeostatic control (negative feedback)
    1. Stimulus– produces a change to a variable (the factor being regulated).
    2. Receptor– detects the change. The receptor monitors the environment and responds to change (stimuli).
    3. Input– information travels along the (afferent) pathway to the control center. The control center determines the appropriate response and course of action.
    4. Output– information sent from the control center travels down the (efferent) pathway to the effector.
    5. Response– a response from the effector balances out the original stimulus to maintain homeostasis.
- Temperature regulation in primates
    1. Sensors detect rise in temperature.

2. Nerve pathways send messages to the control centres in the brain (hypothalamus)
3. Hypothalamus sends signals to sweat glands to produce sweat.
4. Skin cools

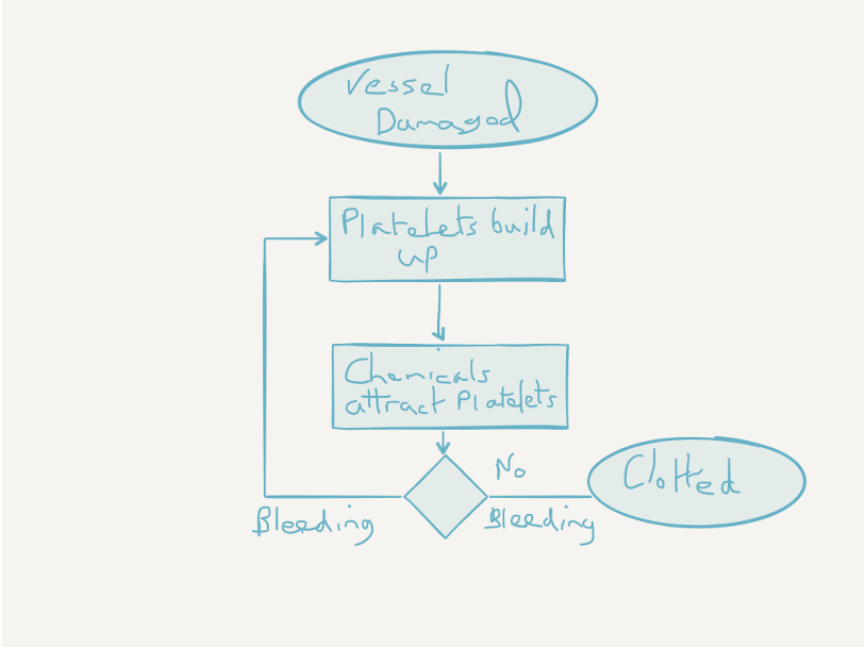
-ve feedback mechanisms



Circuit Breaker on a client.

1. Clients send requests to many requests for a service to handle.
2. Service starts to reject requests.
3. Client implements a circuit breaker to back off until service recovers or another service is spun up to take on requests.

+ve feedback mechanisms

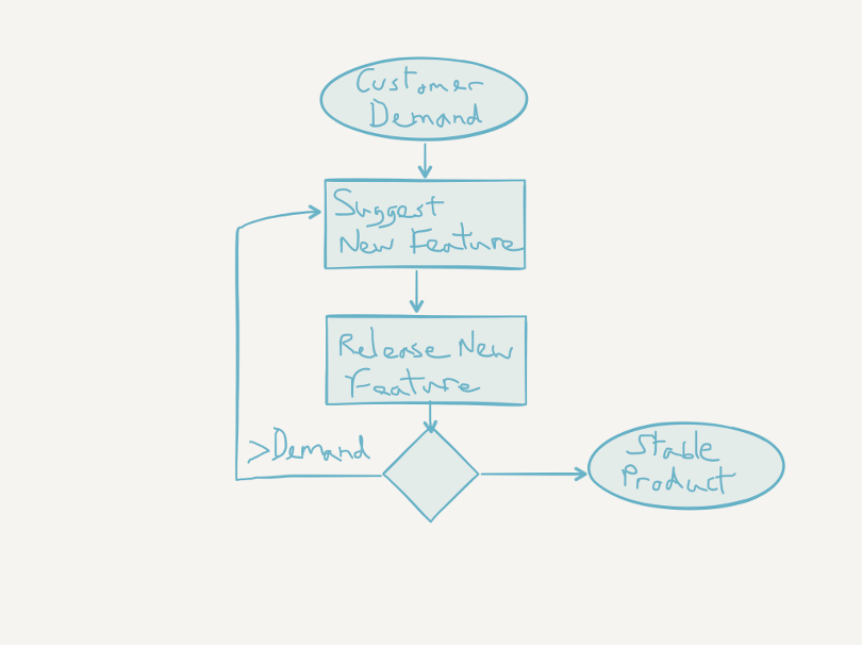


A positive feedback loop comes into play during blood clotting.

1. Once a vessel is damaged,
2. platelets start to cling to the injured site and release chemicals
3. that attract more platelets.

The platelets continue to pile up and release chemicals until a clot is formed.

+ve feedback mechanisms



Positive feedback in software is driven by external agencies such as customer demand, management demand, changes in organisational focus.

Studies have show that too much positive feedback promoting large changes in a single generation (version) of software systems tends to have a destabilising effect if there is not a commensurate negative feedback loop to dampen the effect. This can be seen in the 8 Laws of Software Evolution documented by M.M.Lehman.

Positive feedback loops in software development (evolution) are the introduction of new features, new capabilities, changes in business model resulting in massive changes in structure and capabilities of not just the software design itself but in the structure of the team and the processes adopted by the team in software development (evolution).

This is not always negative - positive feedback introducing changes in development process or materials can provide balanced growth if the changes introduced have inherent constraints that provide a negative feedback loop or dampening effect to appropriate characteristics of the system design or the development process.

As an example, introducing a new programming language that has features that encourage constraints such as immutable data and pure functions

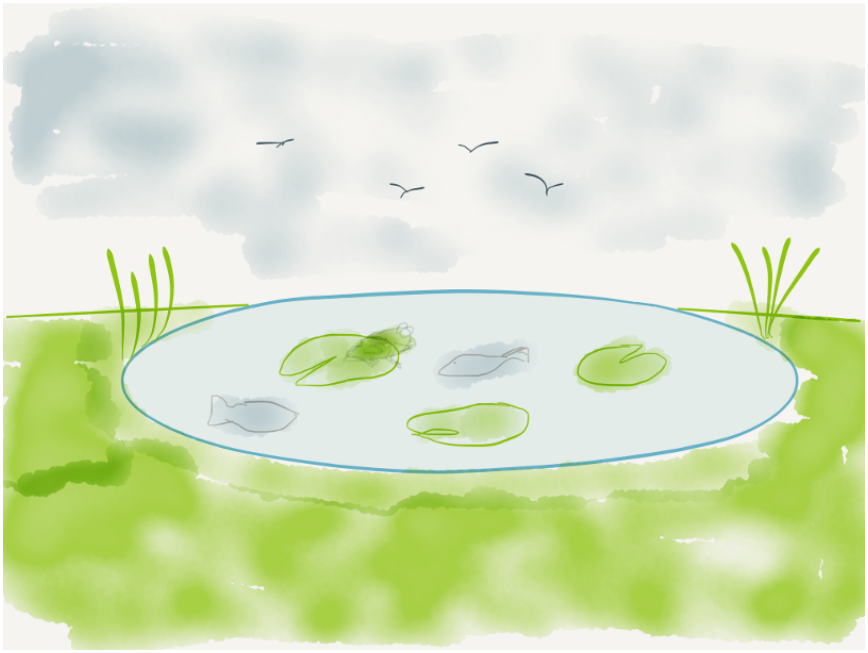
can reduce the complexity of concurrent processing and simplify the developers mental model to not have to consider how state changes over time or race conditions etc. except at the edges of the system/component/service where state changes are pushed i.e. HTTP request/response, I/O to file systems or databases.

Another example of positive feedback in the form of a change to the development process that intentionally introduces a constraint to guide development might be the introduction of WIP limits on a Kanban board. WIP limits ensure the development team are only working at the capacity they can sustain. They reduce the amount of waste in the process in the form of work at rest (waiting to be started). They also highlight when the team has too much work concurrently to manage. WIP limits also make it easier to tune cycle time (time between the work starting and delivery/) and lead time (time between a feature/story/requirement being raised and delivered).

Ecosystems

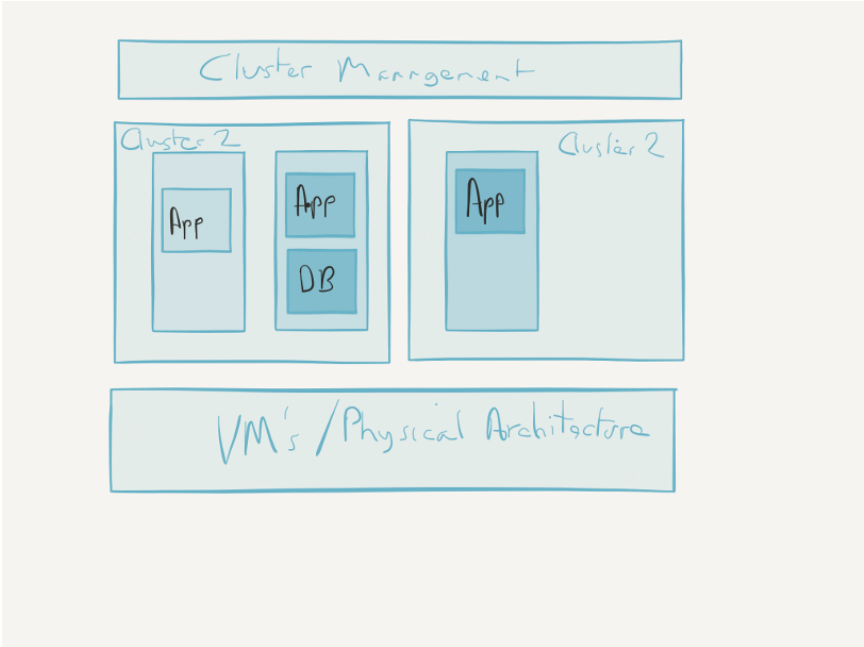
- Different levels of Ecosystem

Micro-ecosystem



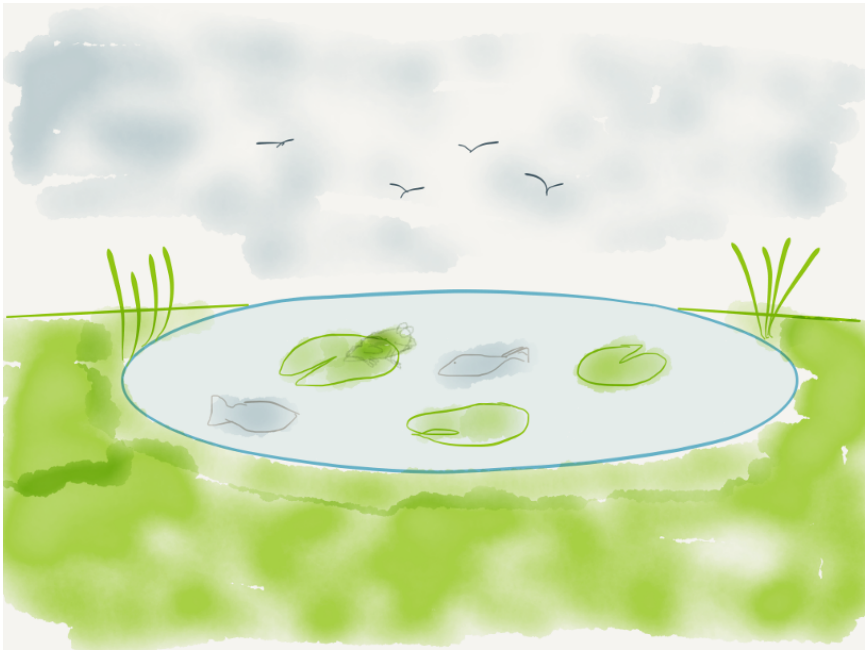
In biology an example of a microecosystem might be a pond which has a limited amount of variation. It's temperature differences are buffered. Water levels can raise or fall but are usually in predictable bounds.

Micro-ecosystem



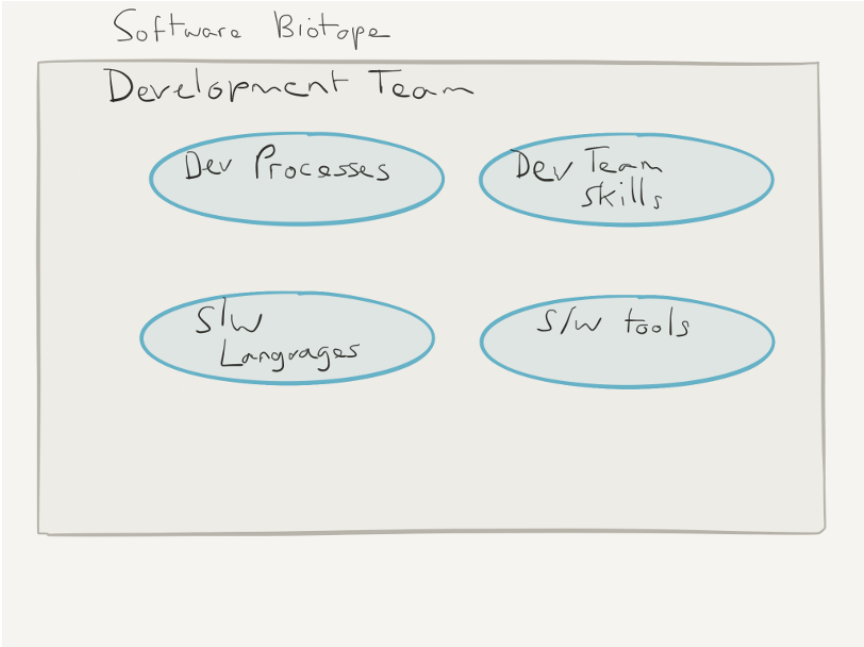
1. Software executables
2. OS
3. Runtime environments
4. Monitoring and logging.
5. etc.

Habitat or Biotope



In biology an example of a habitat might be the garden the pond is in. It has a limited number of species of flora and fauna. Predictable amounts of rainfall, temperature, shade etc.

Habitat or Biotope

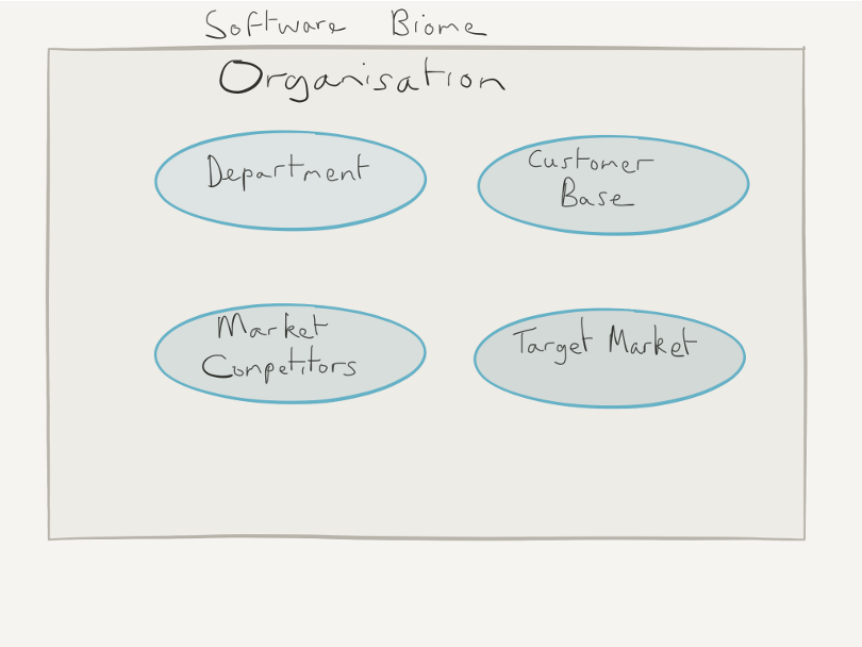


1. Development processes
2. Development team interactions
3. Software languages
4. Software tools

Biome

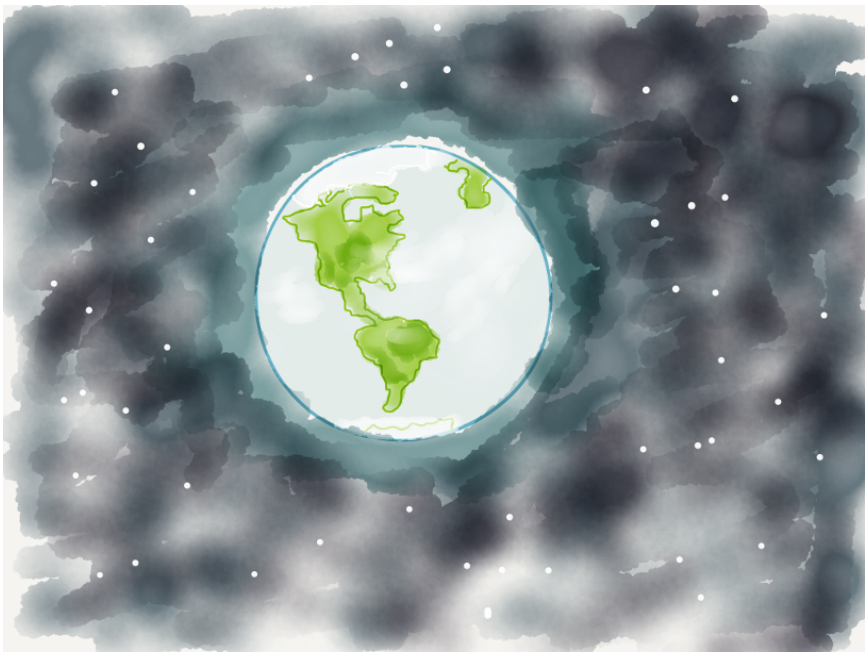
In biology an example of a Biome would be Tropical Rainforest, Tundra, Hot Desert, etc.

Biome



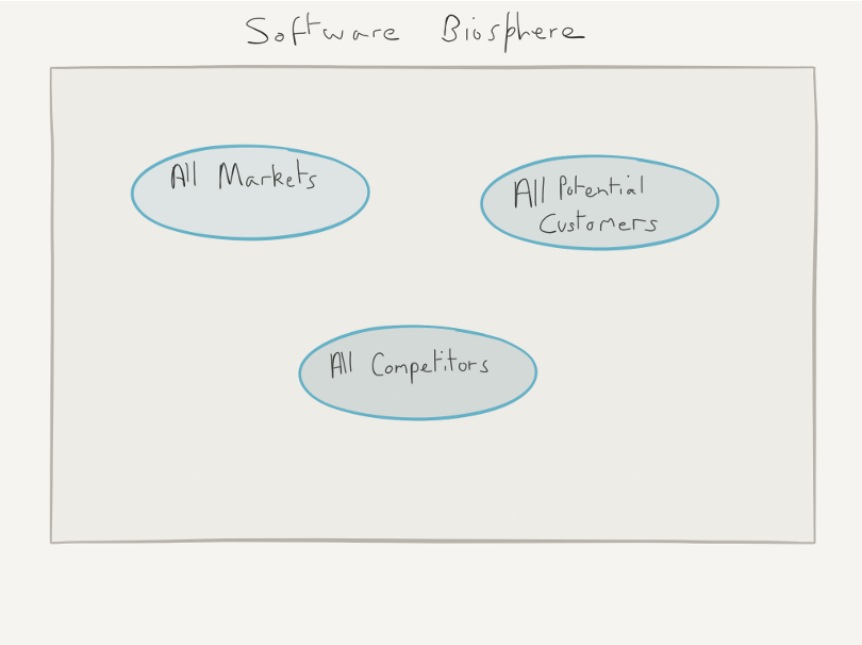
1. Department or organisation.
2. Customer base
3. Target market
4. Competitors in the same market

Biosphere



In biology a biosphere is planetary scale e.g. Earth.

Biosphere



1. All markets
2. All potential customers
3. All competitors

Evolutionary Mechanism

- Adaption
- Acclimatisation
- Other mechanisms

Adaption

- Mechanisms that involve large changes to respond to environment pressures in the ecosystem. For example, a large and permanent temperature change, the introduction on competitive species.

- Adaption involves changes in the phenotype, therefore involving a new generation (version) of the 'species'. For example, adult stature or eye colour.

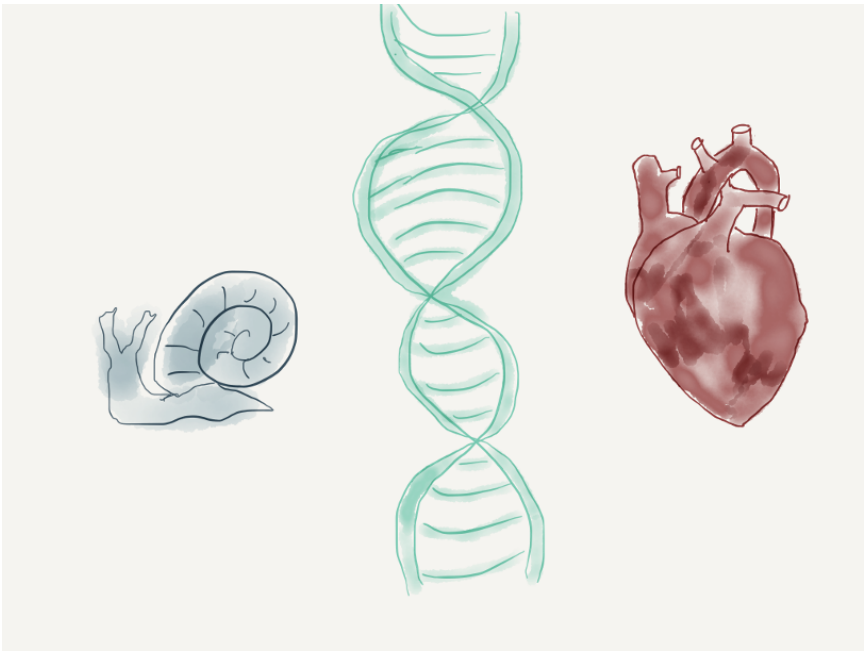
Acclimatisation

- Mechanisms that involve typically smaller variations in phenotypically plastic characteristics. For example, adjustments in heart rate, skin colour and attention span.

Other mechanisms

- there are mechanisms in Epigenetics where external or environmental factors result in a heritable phenotype change or phenotypically plastic changes. For example, stress factors in the environment may trigger a gene to activate or inhibit the expression of a gene without actually changing the nucleotide sequence.

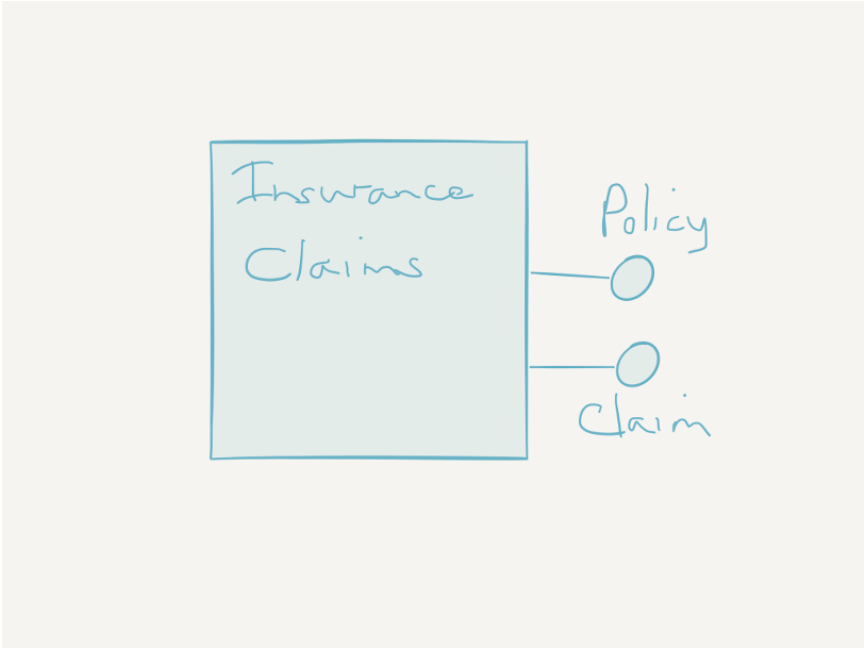
Changes in Biome/Biotope



Biological responses

- Most organisms can't respond to major changes in their Biome by acclimatisation as the small variations possible are usually changes that are only capable of responding to the smaller variations that are confined to the local ecosystem of the habitat or biotope.
- Gradual or smaller changes to the biome can be responded to by adaption that happen in one or two generations through epigenetic mechanisms such as phenotype plasticity. These might be something like a dormant gene being activated by external factors. Morphological plasticity can be seen in pond snails that develop spiny shells in the presence of predators.
- Larger changes in the biome require more major genetic changes in the DNA and occur through a number of mechanisms, the best understood of which is gene mutation but recent research suggests other mechanisms such as stress hormones altering RNA transcription and the adoption of phenotypically plastic traits.

Changes in Biome/Biotope



Software response

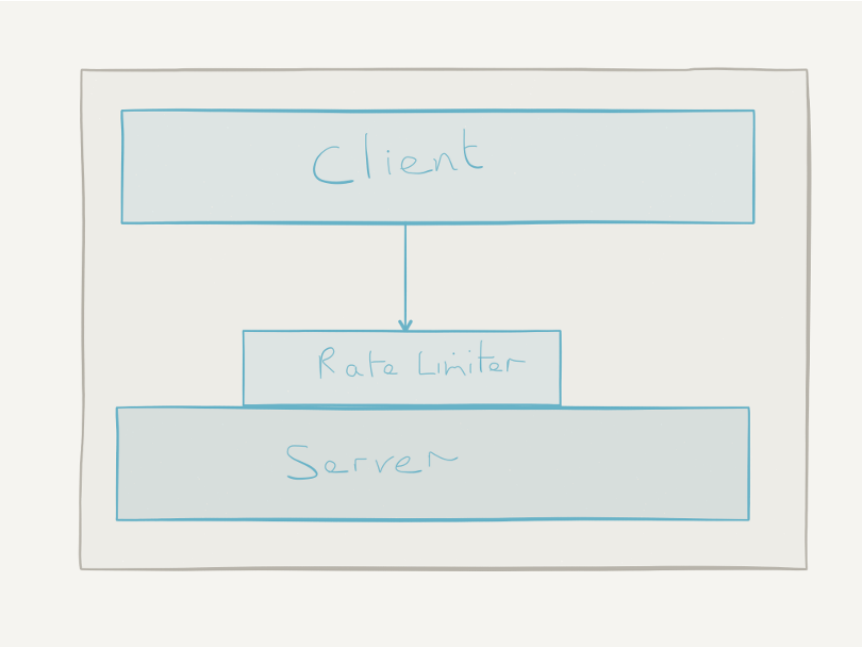
- The possible response mechanisms in software are controlled by the development team and associated stakeholders. The team can adopt techniques that provided a 'guided' evolutionary mechanisms.
- Smaller changes can be made in days in one generation (version) either by tweaking parameterised features or making small code changes to the internal mechanisms in a service or application. Generally speaking making these changes quickly is much easier if the software component is smaller.
- Larger changes to response take more major changes internally but also tend to change the interfaces of the software component. I.e. the interface for clients or the persistent data used is changed. These changes tend to alter factors that a coupled to other software components or to external customers/systems.
- What is frequently overlooked is that often this kind of change has an impact on the more complex microecosystem. As an example, making a change to add something to an API changes the responsibilities of the software component involved in a way that may mean that the component is now taking on responsibilities that it didn't have before. This can result in other software components at a later time connecting to the first to use this new responsibility. This kind of network effect increase the amount of communication and coordination required and hence may increase cross network traffic and result in hot-spots in the architecture.

Feedback loops

Feedback loops in Biology

- Smaller variations - behavioural changes, physiological changes, phenotypically plastic traits
- Genetic mutation, cooperative strategies.
- Different species move into ecosystem, most drastically extinction.

Feedback loops



Feedback loops in Code

- In general mechanisms to respond to stimuli that are built into 'code' or even 'configuration' tend to be negative feedback loops that respond to a an external stimuli by damping down or stabilising the software system.
- Good examples of this would be circuit breakers, bulkhead patterns or rate limiting.

Feedback loops in Development team and process

- The development team and the processes and tools that they use are a mix of negative feedback loops and positive feedback loops. Generally speaking, these changes are adopted to facilitate faster changes in certain directions. However, conversely, often the approaches and tools are successful in this because they have constraints (or negative feedback) that provides guide rails that push development towards the desired characteristics.
- As examples:

1. adopting Kanban properly can greatly increase the speed of response to changes in the habitat but Kanban as a process has at it's heart strong constraints on the amount of work in progress at any point in time.
2. adopting a functional programming language with immutable data structures can reduce the effort involved in developing software that has concurrent processes.
3. adopting a microservices architecture that constrains the responsibility of a service to just one small part of a domain can result in the ability to respond faster to changes as a microservice can be changed in hours or replaced in days or a week.

Guided Evolution

Development Team are the primary evolutionary mechanism

In order to guide software evolutionary processes the team needs to understand the ecosystem they are designing for. Traditionally, this has been an exercise in gathering and fixing requirements/features/stories but this is often flawed as to accurately determine what the Biotope is using static analysis based on peoples interpretation is inherently unconsciously biased.

In order to determine what the ecosystem is we need to 'sense' the environment we are in.

Sensing micro-ecosystem

- Traditional monitoring of running software. Memory, CPU, etc.
- Monitoring and alerting on network effects like request rates, average/std deviation of request latency, etc.

Sensing the Habitat/Biotope

- Dwell time on a site
- Conversion rates
- Sales rates
- Return rates

- Usage rates of features
- Customer location
- Customer segmentation
- A/B testing
- Scientist type code experiments. Code that performs old and new way, compares results and reports differences while taking the old result as response.

Sensing the Biome

- Surveying current customers and wider customer base
- Market research
- Industry research
- Exploratory features or new apps targeting other markets
- P & L
- Board level decisions

Sensing the Biosphere

- Major organisational events (mergers, buyouts, bankruptcy proceedings)

Feedback from sensors

- tune evolution
- Feedback from these sensors are used to tune the evolutionary processes at the appropriate level.
- You can move to respond to changes in a wider ecosystem by changing the processes and the sensors to guide evolution to respond to the large change over generations of the software design.
- Conversely, detecting changes in a sensors output over time in a narrower ecosystem can suggest that a large change is occurring in a wider ecosystem.

Factors that influence software evolution

- Not designing a sensor/response feedback mechanism
- Bad development practices making response to change harder or impossible
  - Poor code hygene
- Constraints on the ecosystems that inhibit the ability to sense or respond to change.
  - Typically, edicts to follow a directive/std with no understanding or explanation of the reasoning behind this.
- Over population effects poisoning the ecosystem.

Conclusion

- Identify the factors that matter and pick metrics that measure them
- Design your feedback loops (sensor/receptors) and effectors
- Identify beneficial constraints vs detrimental constraints
  - e.g. service boundaries or immutable data vs inability to change biotope due to policy/stds
  - Ensure Ecosystems are self contained
  - Constrain & understand changes from upper ecosystem.
- People (the team) are your mechanism for evolution - they need to know what constraints are imposed and why and the directions of freedom as well as the results of sensory input.

Citations

- Positive feedback and alternative stable states in inbreeding, cooperation, sex roles and other evolutionary processes - Jussi Lehtonen and Hanna Kokko
- Epigenetic Feedback Regulation Accelerates Adaptation and Evolution - Chikara Furusawa, Kunihiko Kaneko

- A feedback model of evolutionary adaptation - Daniele Casagrande, Wiesław Krajewski, Umberto Viaro
- Differential Evolution - Clever Algorithms: Nature-Inspired Programming Recipes - Jason Brownlee PhD
- On the nature of change - Brendan Montague
- Listening to nature: How sound can help us understand environmental change - Garth Paine
- How can a Biological System be affected by a change in pH levels - JacobS
- Fold-change detection in biological systems - MiriAdler, UriAlon
- On the relation between fluctuation & response in biological systems - Katsuhiko Sato, Yoichiro Ito, Tetsuya Yomo, and Kunihiko Kaneko
- DNA and Knot Theory - M. Beals, L. Gross, S. Harrell
- 6 Animals that are Rapidly Evolving - Jessica Hullinger
- 5 Examples of How Organisms respond to Stimuli - Austin Brashear Brashear
- Cells can respond to changes in their environment - Berg JM, Tymoczko JL, Stryer L
- A Guide to Neurotransmitter Balance - `https://poweronpoweroff.com`
- Difference between Adaptation and Acclimation - maureen. Difference-Between.net. March 15, 2011
- Animal Sensory Systems - Shana Kerr. November 13, 2016
- Microservices, biology and evolution - Sajid Kalla. medium.com
- The role of behaviour in evolution - Prof Adrian Lister, Prof Ian Barnes. Natural History Museum London
- How do ants assess food volume? - Anne-Catherine Mailleux, Jean-Louis Deneubourg & Claire Detrain

- Laws of Software Evolution Revisited - M.M.Lehman
- Building Evolutionary Architectures - Neal Ford, Rebecca Parsons & Patrick Kua
- Domain Driven Design Distilled - Vaughn Vernon