Abstract & Patterns

The acts of the mind, wherein it exerts its power over simple ideas, are chiefly these three:

1. *Combining several simple ideas into one compound one, and thus all complex ideas are made.*
2. *The second is bringing two ideas, whether simple or complex, together, and setting them by one another so as to take a view of them at once, without uniting them into one, by which it gets all its ideas of relations.*
3. *The third is separating them from all other ideas that accompany them in their real existence: this is called abstraction, and thus all its general ideas are made.*

John Locke, An Essay Concerning Human Understanding (1690)

https://mitpress.mit.edu/sicp/full-text/book/book-Z-H-9.html#%_chap_1

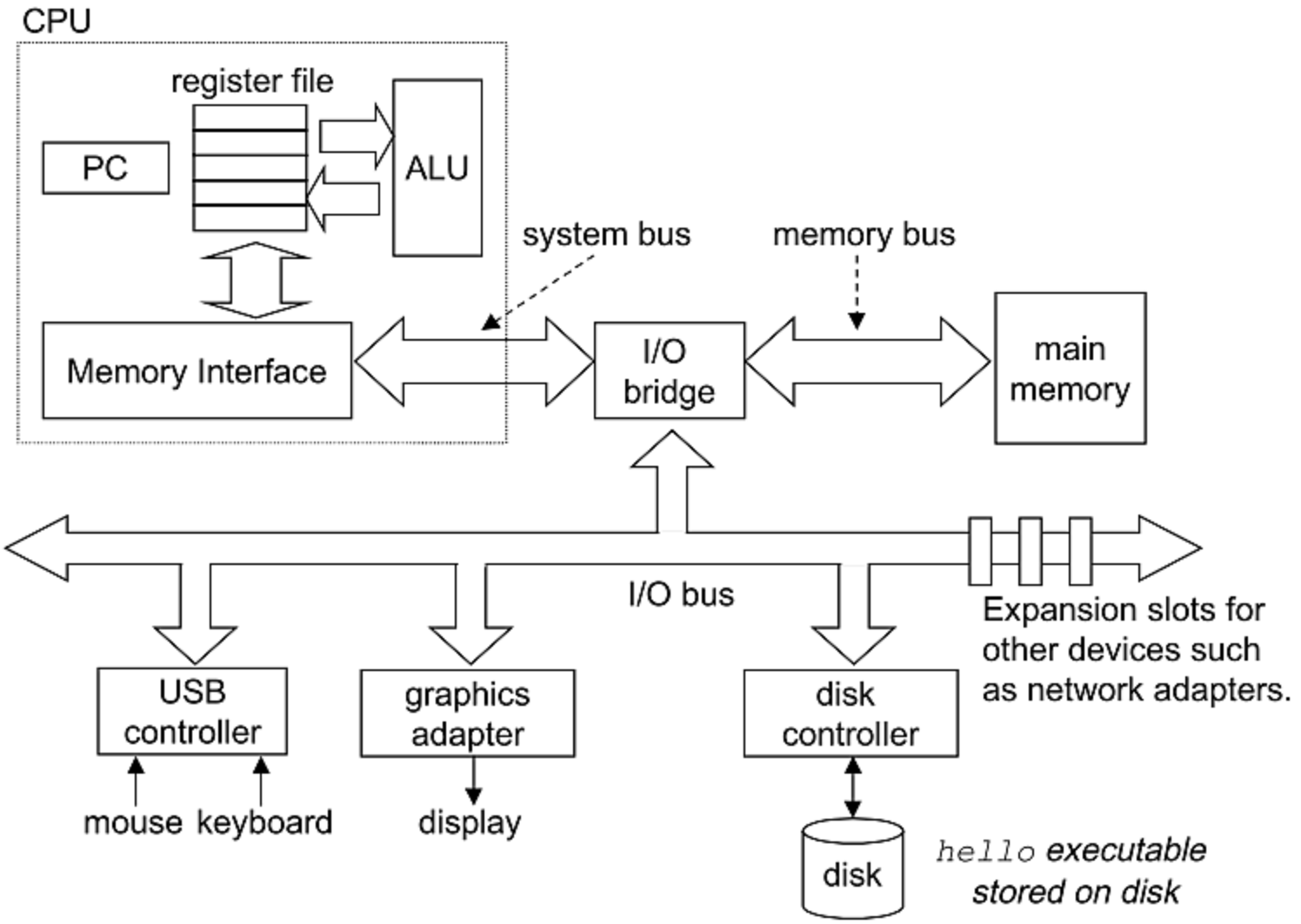
KISS

Keep It Simple, Stupid!

Simplicity is prerequisite for reliability

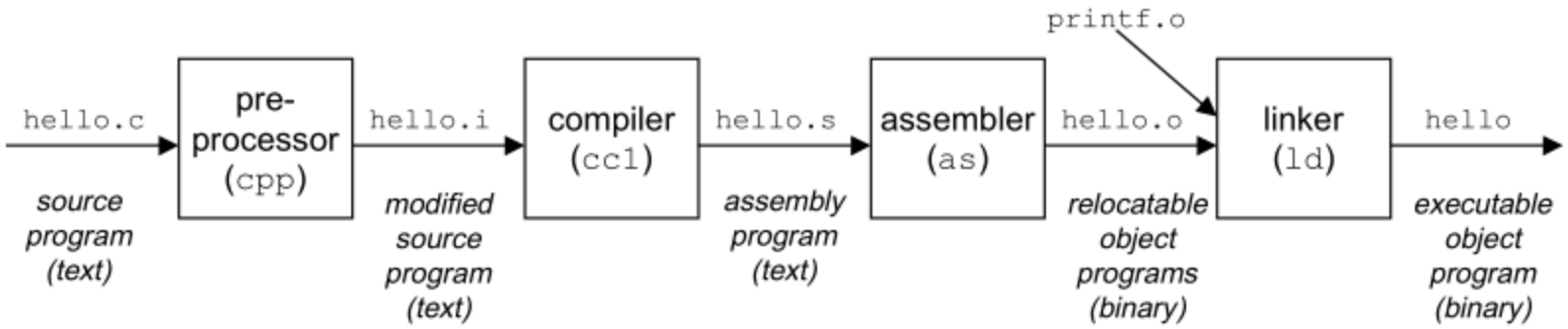
Edsger W. Dijkstra

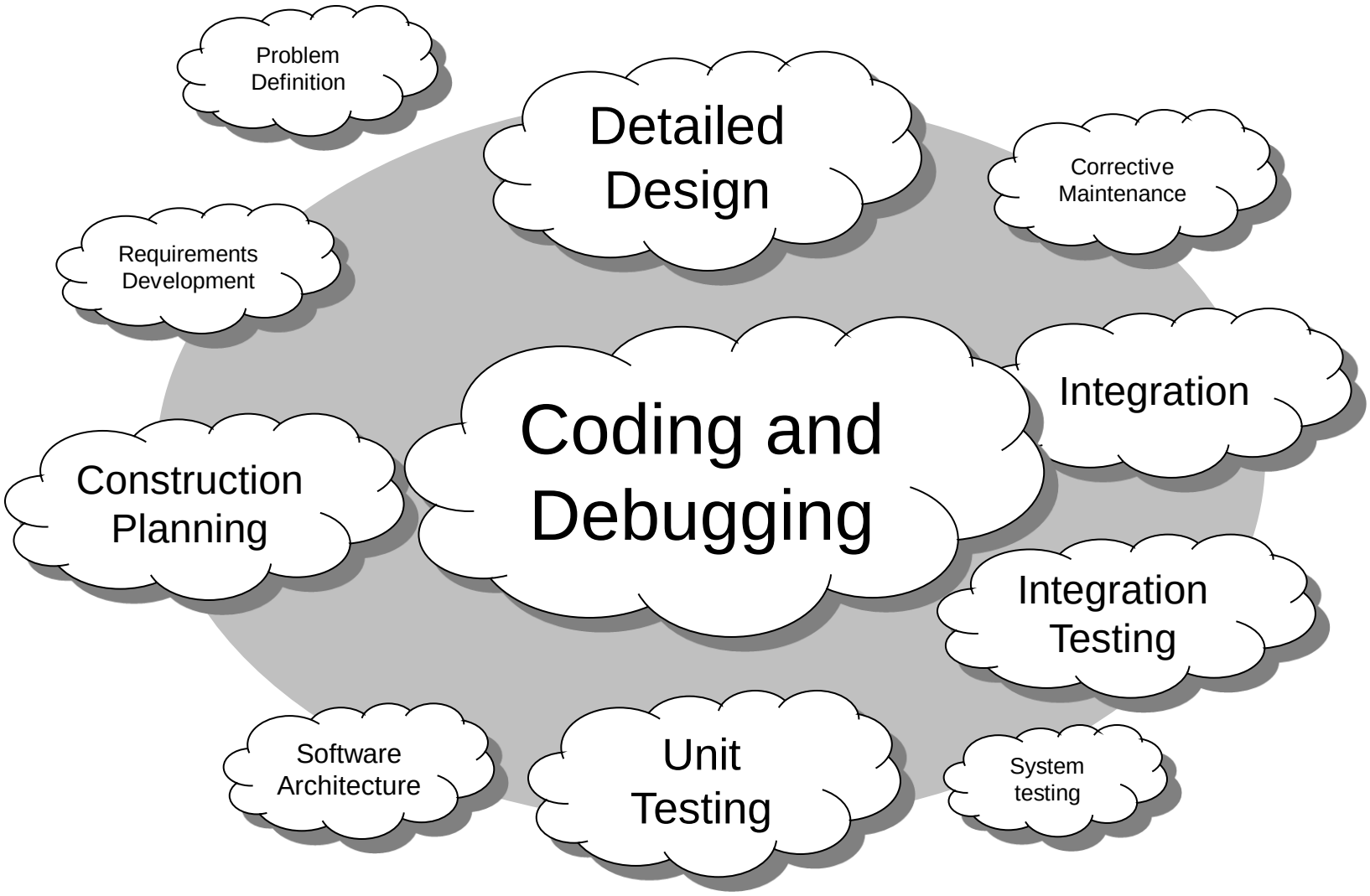
https://en.wikiquote.org/wiki/Edsger_W._Dijkstra


CPU
register file
PC
ALU
system bus
memory bus
Memory Interface
I/O bridge
main memory
I/O bus
Expansion slots for other devices such as network adapters.
USB controller
graphics adapter
disk controller
mouse
keyboard
display
disk
hello executable stored on disk

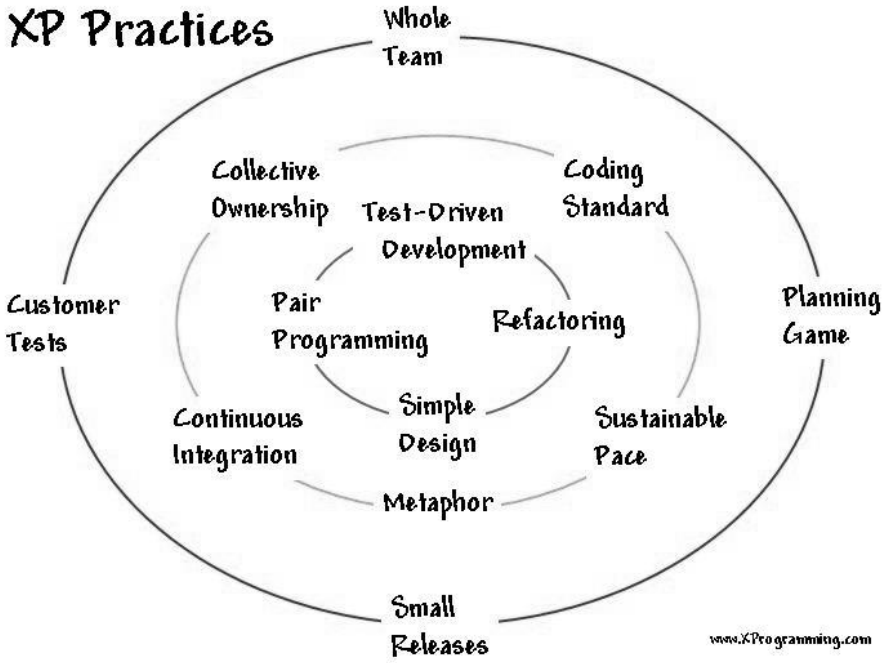
```
#include <stdio.h>

int main()
{
   printf("hello, world\n");
}
```





http://cc2e.com/



Manifesto for Agile Software Development

- *Individuals and interactions over processes and tools*
- *Working software over comprehensive documentation*
- *Customer collaboration over contract negotiation*
- *Responding to change over following a plan*

http://www.agilemanifesto.org/

Complexity	Simplicity
State, Objects	Values
Methods	Functions, Namespaces
Vars	Managed refs
Inheritance, switch, matching	Polymorphism a la carte
Syntax	Data
Imperative loops, fold	Set functions
Actors	Queues
ORM	Declarative data manipulation
Conditionals	Rules
Inconsistency	Consistency

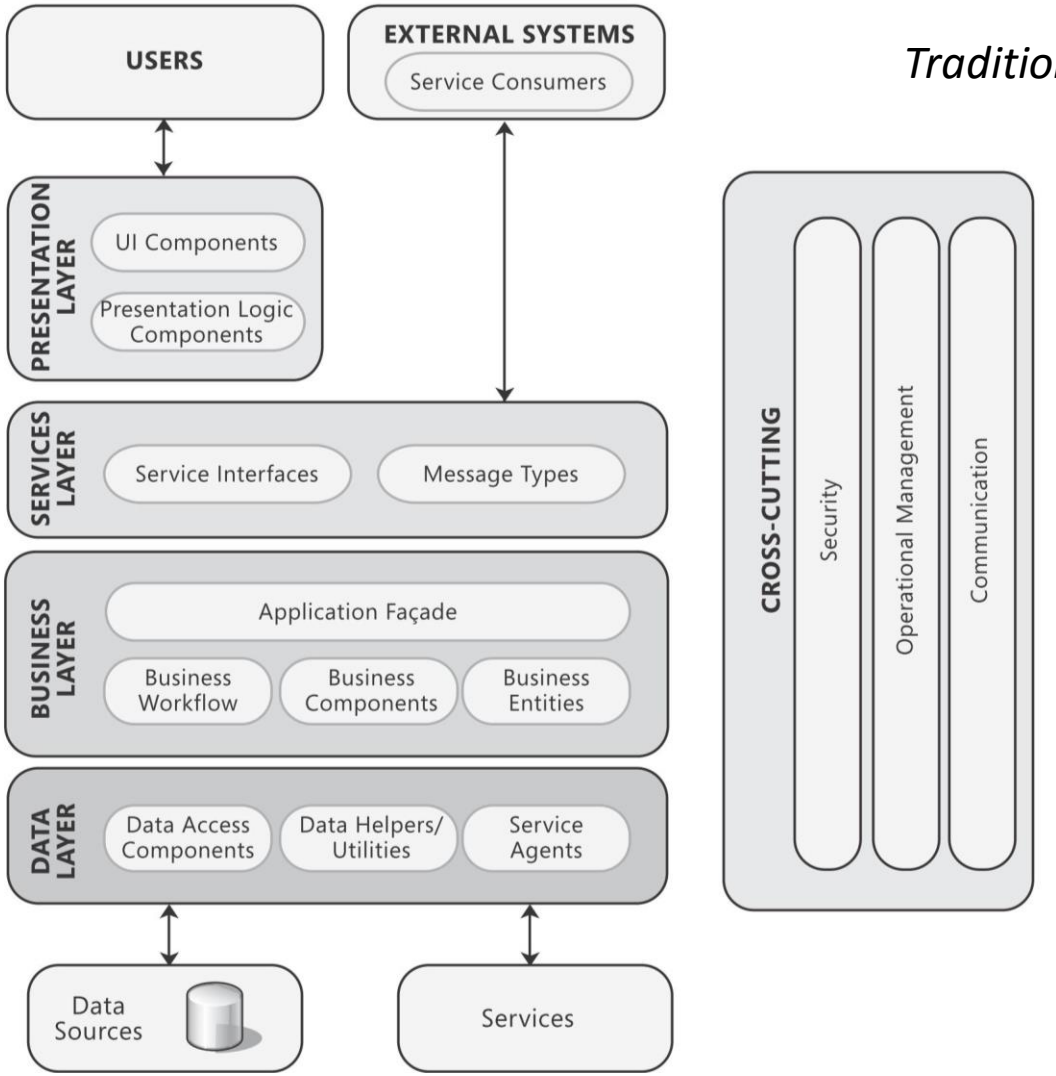
Simplicity Made Easy

- *Choose simple constructs over complexity-generating constructs.*
- *Create abstractions with simplicity as a basic.*
- *Simplify the problem space before you start.*
- *Simplicity often means making more things, not fewer.*
- *Reap the benifits!*

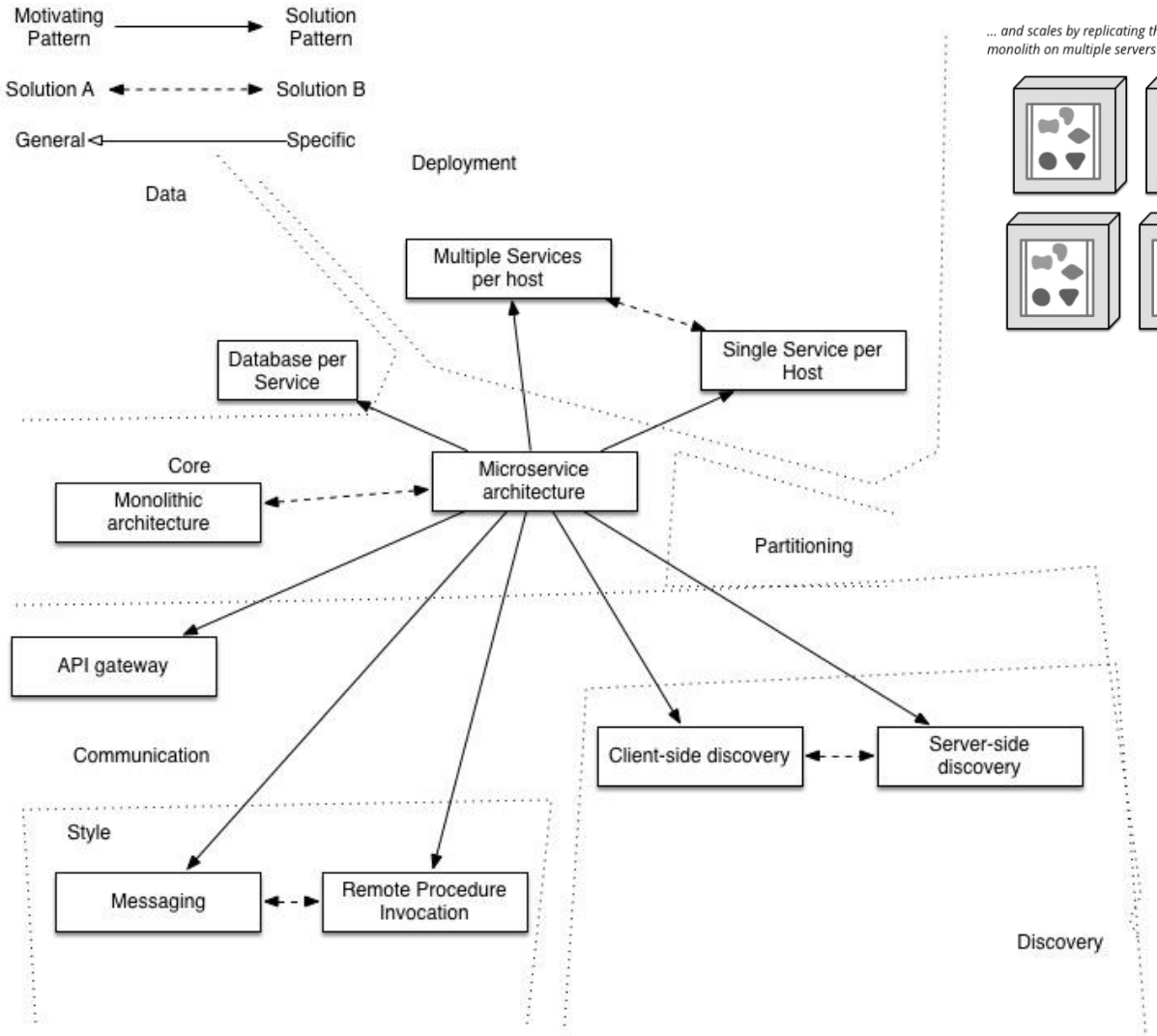
Rich Hickey

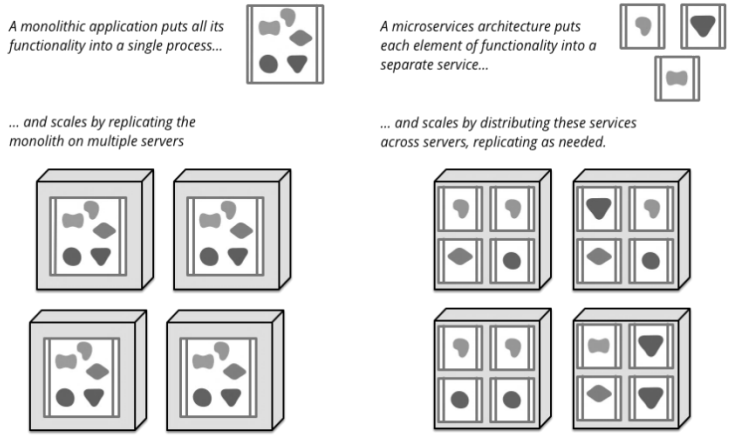
https://github.com/matthiasn/talk-transcripts/blob/master/Hickey_Rich/SimpleMadeEasy.md

Traditional Application Architecture



https://msdn.microsoft.com/en-us/library/ff650706.aspx

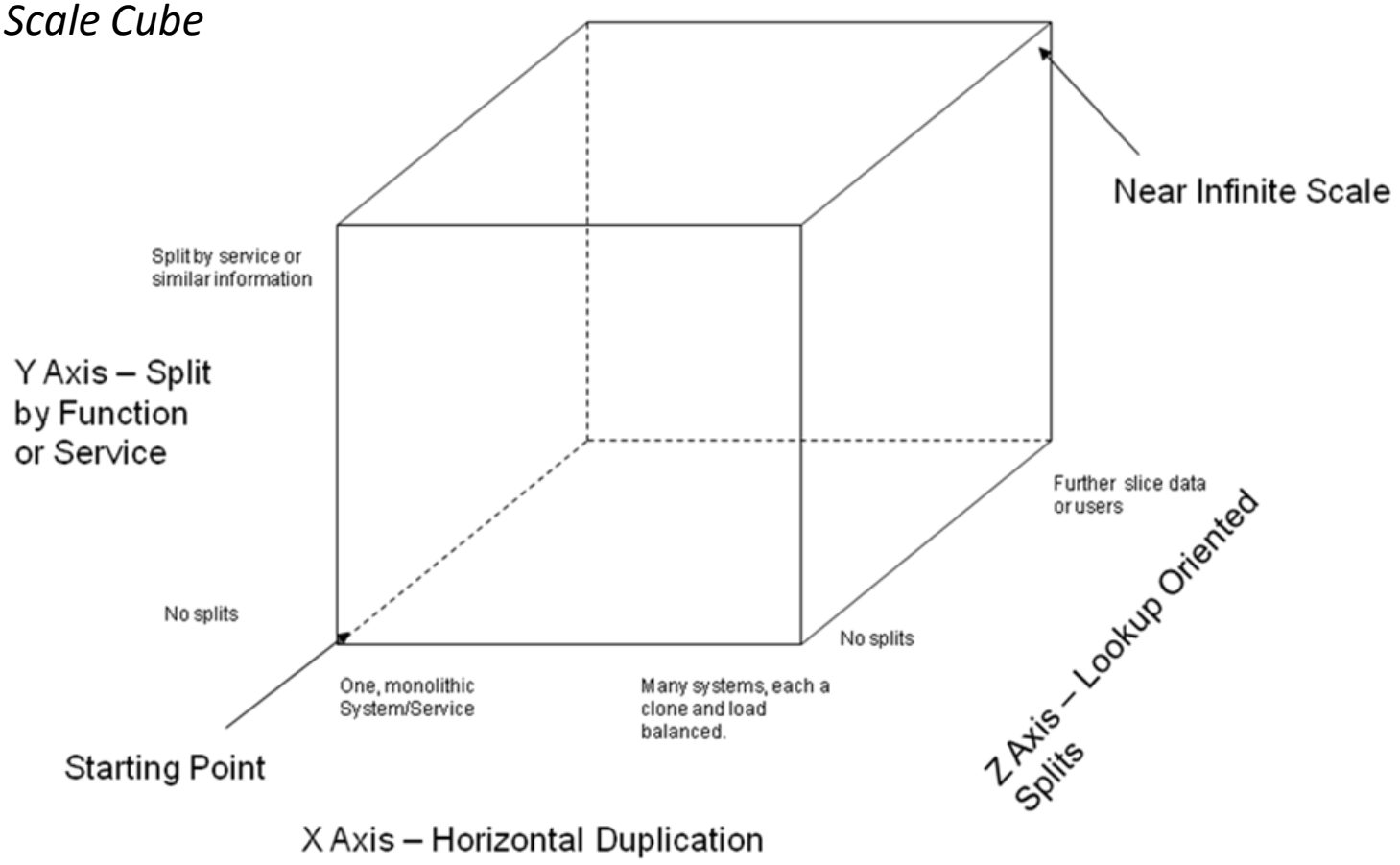




http://martinfowler.com/articles/microservices.html

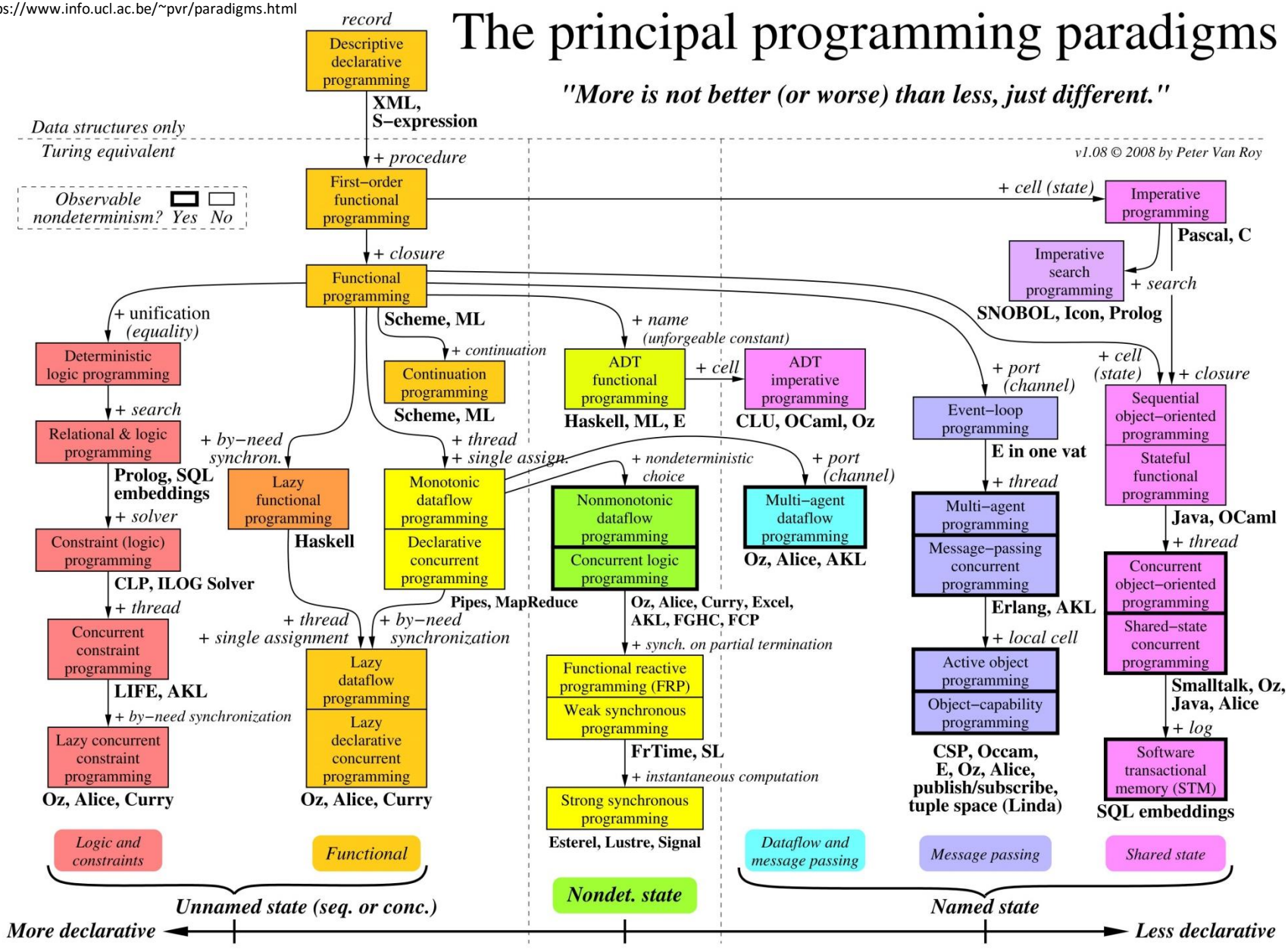
http://microservices.io/patterns/microservices.html

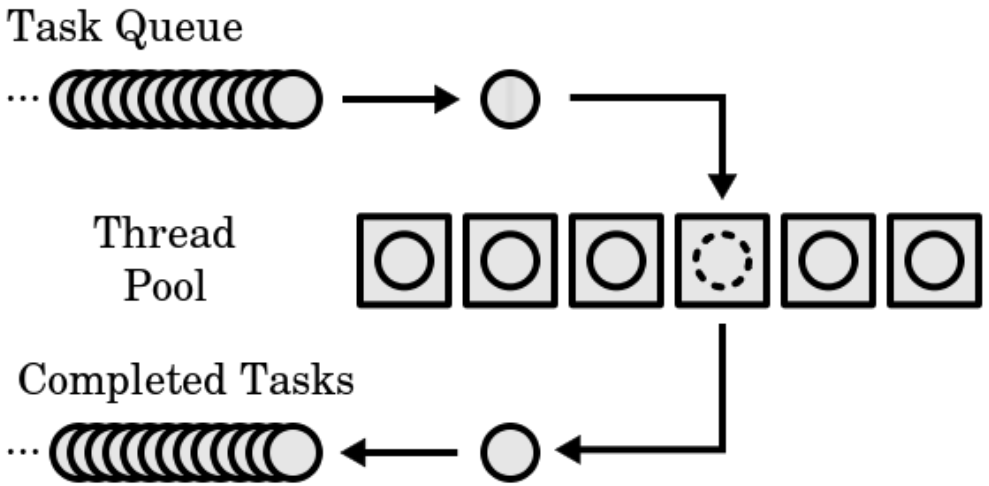
Scale Cube



http://akfpartners.com/techblog/2008/05/08/splitting-applications-or-services-for-scale/

https://www.info.ucl.ac.be/~pvr/paradigms.html





Thread

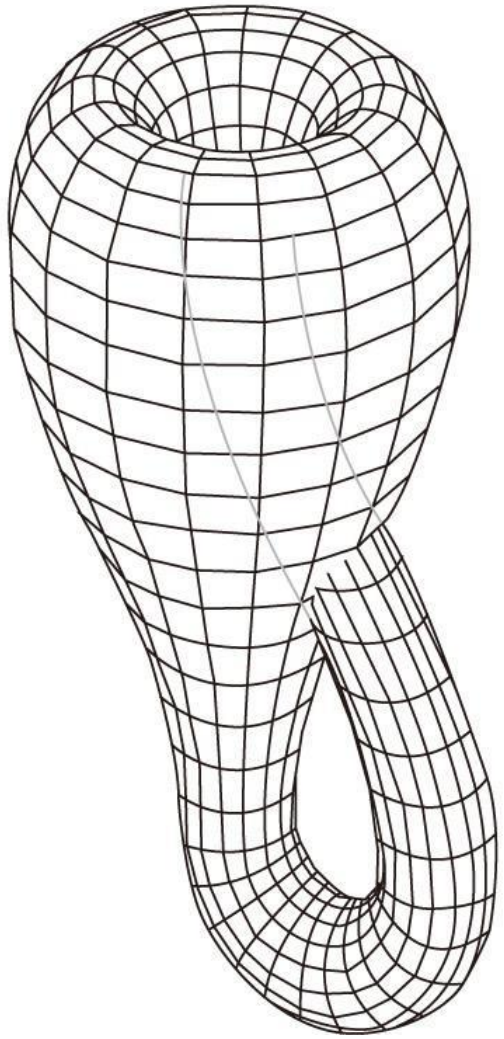
In computer science, a thread of execution is the smallest sequence of programmed instructions that can be managed independently by a scheduler, which is typically a part of the operating system.[1] The implementation of threads and processes differs between operating systems, but in most cases a thread is a component of a process. Multiple threads can exist within one process, executing concurrently (one starting before others finish) and share resources such as memory, while different processes do not share these resources. In particular, the threads of a process share its instructions (executable code) and its context (the values of its variables at any given time).

https://en.wikipedia.org/wiki/Thread_(computing)

Closure

In programming languages, closures (also lexical closures or function closures) are a technique for implementing lexically scoped name binding in languages with first-class functions. Operationally, a closure is a record storing a function together with an environment: a mapping associating each free variable of the function (variables that are used locally, but defined in an enclosing scope) with the value or storage location to which the name was bound when the closure was created. A closure—unlike a plain function—allows the function to access those captured variables through the closure's reference to them, even when the function is invoked outside their scope.

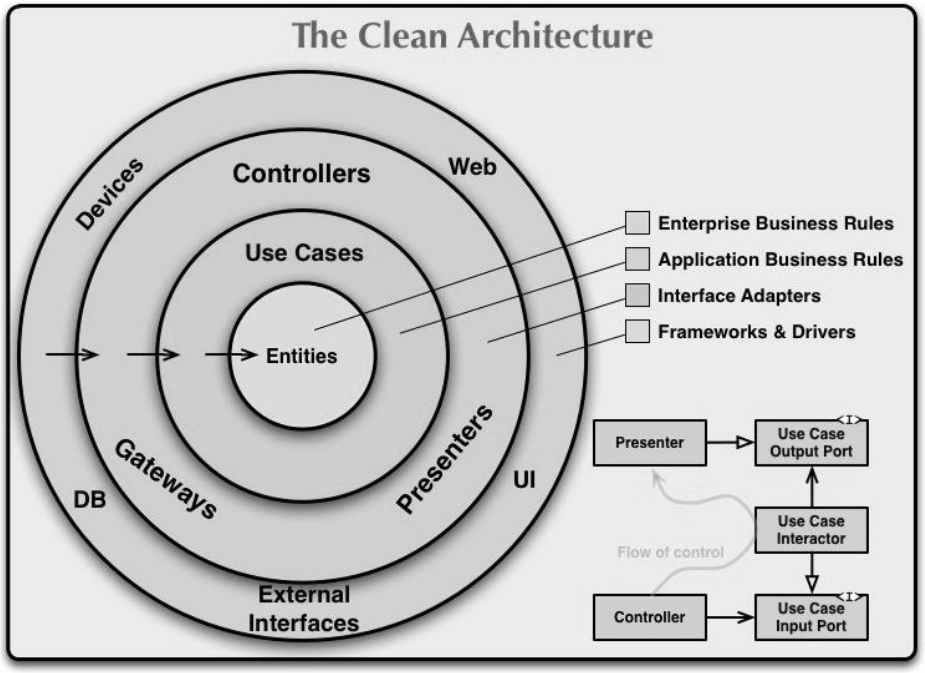
https://en.wikipedia.org/wiki/Closure_(computer_programming)



Software design pattern

In software engineering, a design pattern is a general reusable solution to a commonly occurring problem within a given context in software design. A design pattern is not a finished design that can be transformed directly into source or machine code. It is a description or template for how to solve a problem that can be used in many different situations. Patterns are formalized best practices that the programmer can use to solve common problems when designing an application or system.

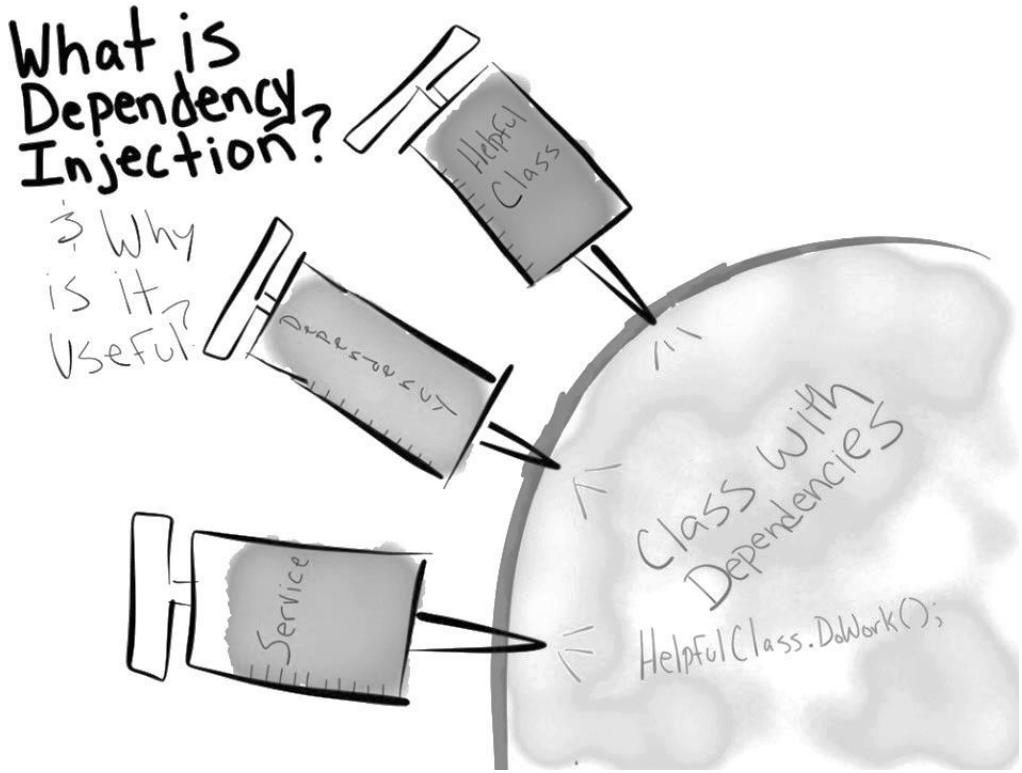
https://en.wikipedia.org/wiki/Software_design_pattern



Separation of concerns

In computer science, separation of concerns (SoC) is a design principle for separating a computer program into distinct sections, such that each section addresses a separate concern. A concern is a set of information that affects the code of a computer program. A concern can be as general as the details of the hardware the code is being optimized for, or as specific as the name of a class to instantiate. A program that embodies SoC well is called a modular program. Modularity, and hence separation of concerns, is achieved by encapsulating information inside a section of code that has a well-defined interface. Encapsulation is a means of information hiding. Layered designs in information systems are another embodiment of separation of concerns.

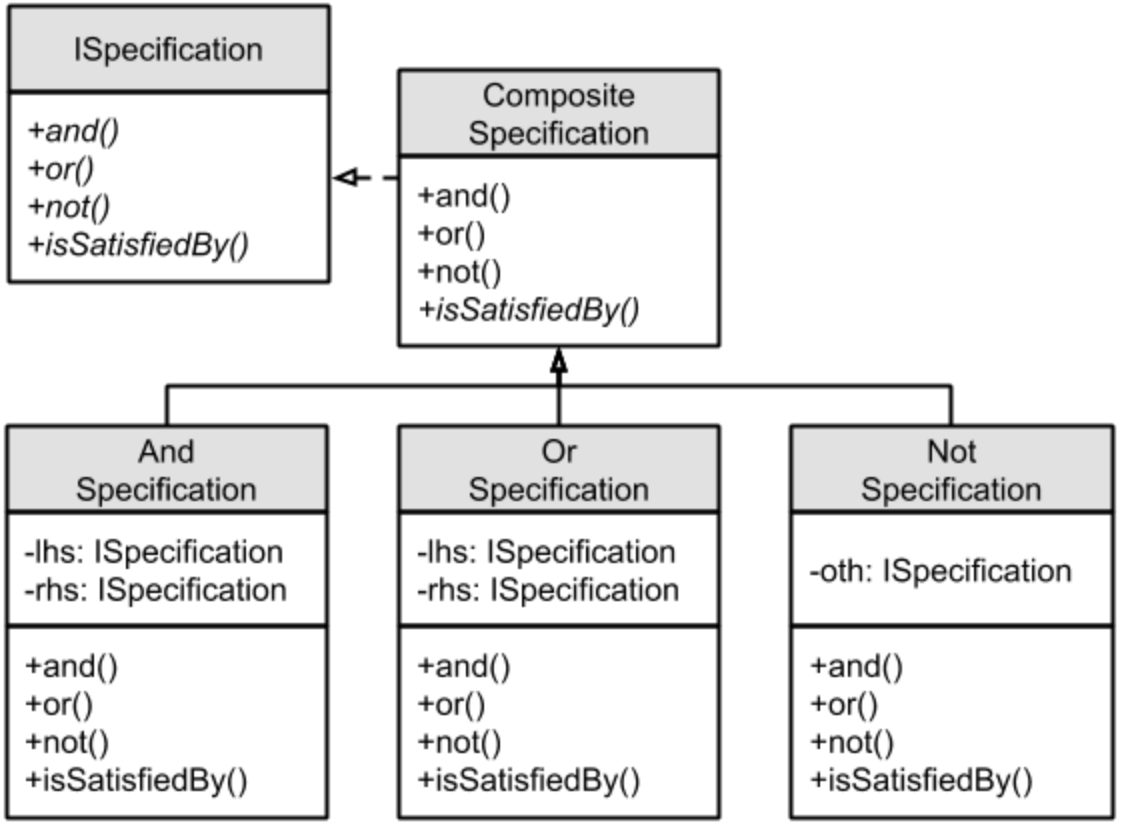
https://en.wikipedia.org/wiki/Separation_of_concerns#SoC_via_partial_classes_in_C.23



Dependency injection

In software engineering, dependency injection is a software design pattern that implements inversion of control for resolving dependencies. A dependency is an object that can be used (a service). An injection is the passing of a dependency to a dependent object (a client) that would use it. The service is made part of the client's state. Passing the service to the client, rather than allowing a client to build or find the service, is the fundamental requirement of the pattern.

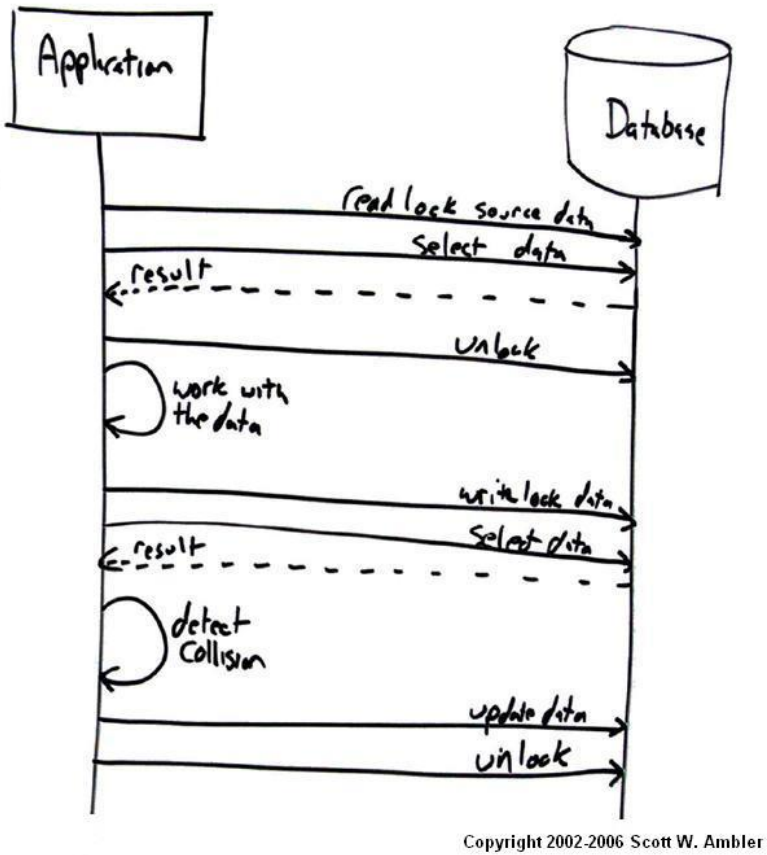
https://en.wikipedia.org/wiki/Dependency_injection



Specification pattern

In computer programming, the specification pattern is a particular software design pattern, whereby business rules can be recombined by chaining the business rules together using boolean logic. The pattern is frequently used in the context of domain-driven design.

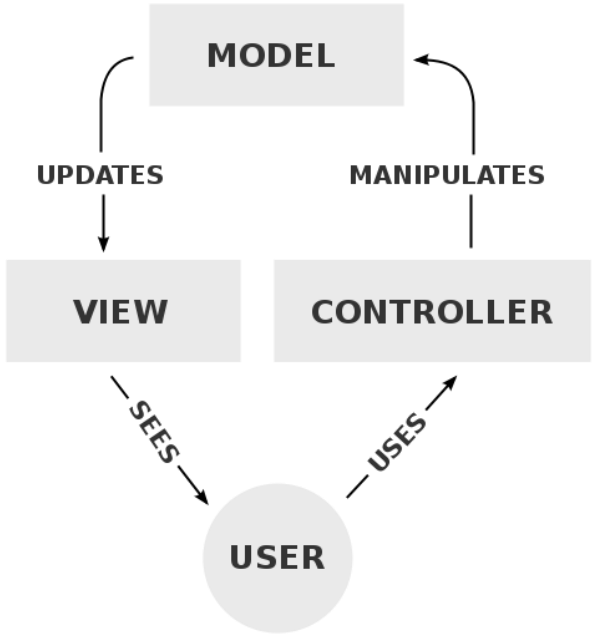
https://en.wikipedia.org/wiki/Specification_pattern



Lock

In computer science, a lock or mutex (from mutual exclusion) is a synchronization mechanism for enforcing limits on access to a resource in an environment where there are many threads of execution. A lock is designed to enforce a mutual exclusion concurrency control policy.

https://en.wikipedia.org/wiki/Lock_(computer_science)



Model–view–controller

Model–view–controller (MVC) is a software architectural pattern mostly (but not exclusively) for implementing user interfaces on computers. It divides a given software application into three interconnected parts, so as to separate internal representations of information from the ways that information is presented to or accepted from the user.

https://en.wikipedia.org/wiki/Model-view-controller

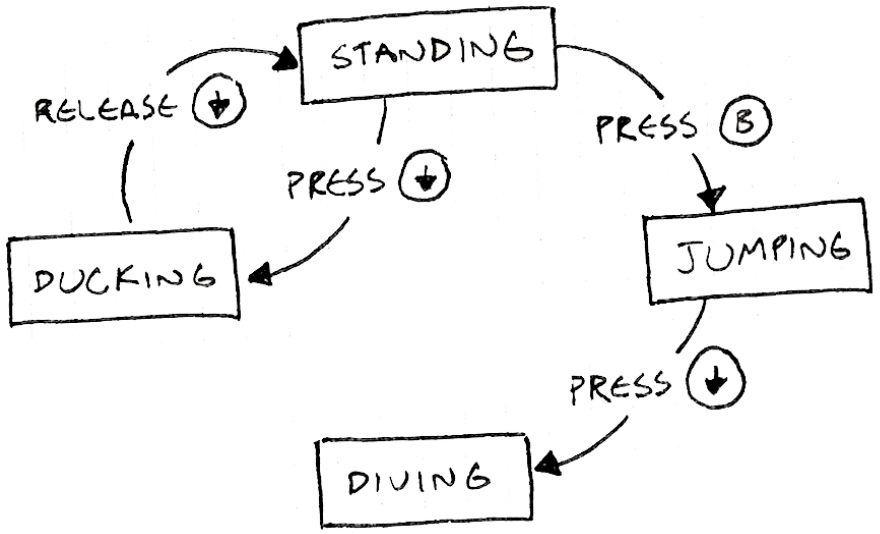
	C++	SML	OCaml	Haskell	Java	C#	Cecil	C++0X	G	JavaGI	Scala
Multi-type concepts	-	●	○	●	●[2]	●[2]	◐	●	●	●	●[2]
Multiple constraints	-	◐	◐	●	●	●	●	●	●	●	●
Associated type access	●	●	◐	●	◐	◐	◐	●	●	◐	●[1]
Constraints on assoc. types	-	●	●	●	◐	◐	●	●	●	◐	●[1]
Retroactive modeling	-	●	●	●	◐[2]	◐[2]	●	●	●	●	●[23]
Type aliases	●	●	●	●	○	○	○	●	●	○	●
Separate compilation	○	●	●	●	●	●	◐	○	●	●	●
Implicit arg. deduction	●	○	●	●	◐[5]	◐[5]	◐	●	●	●	●[3]
Modular type checking	○	●	◐	●	●	●	◐	◐	●	◐	●
Lexically scoped models	○	●	○	○	○	○	○	○	●	○	●
Concept-based overloading	●	○	○	○	○	○	●	●	◐	○	◐[4]
Equality constraints	-	●	○	●	○	○	○	●	●	○	●
First-class functions	○	●	●	●	○	◐	●	●	◐	○	●

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.176.5300&rep=rep1&type=pdf

Generic programming

In the simplest definition, generic programming is a style of computer programming in which algorithms are written in terms of types to-be-specified-later that are then instantiated when needed for specific types provided as parameters. This approach, pioneered by ML in 1973, permits writing common functions or types that differ only in the set of types on which they operate when used, thus reducing duplication.

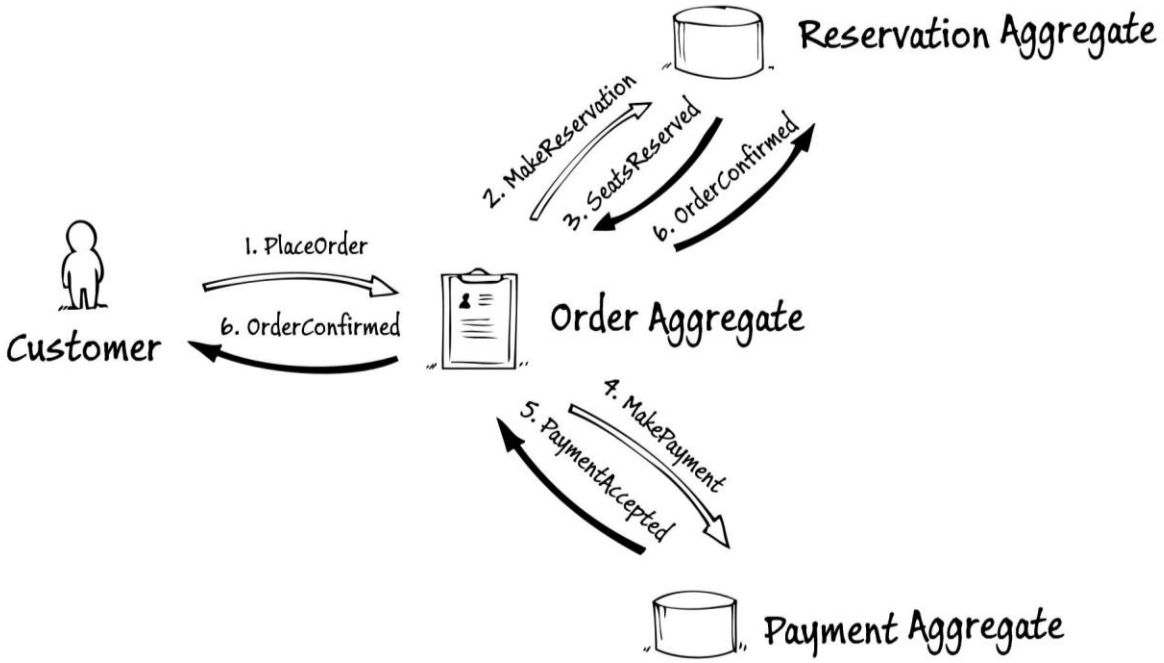
https://en.wikipedia.org/wiki/Generic_programming



Finite-state machine

A finite-state machine (FSM) or finite-state automaton, or simply a state machine, is a mathematical model of computation used to design both computer programs and sequential logic circuits. It is conceived as an abstract machine that can be in one of a finite number of states. The machine is in only one state at a time; the state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition; this is called a transition. A particular FSM is defined by a list of its states, and the triggering condition for each transition.

https://en.wikipedia.org/wiki/Finite-state_machine



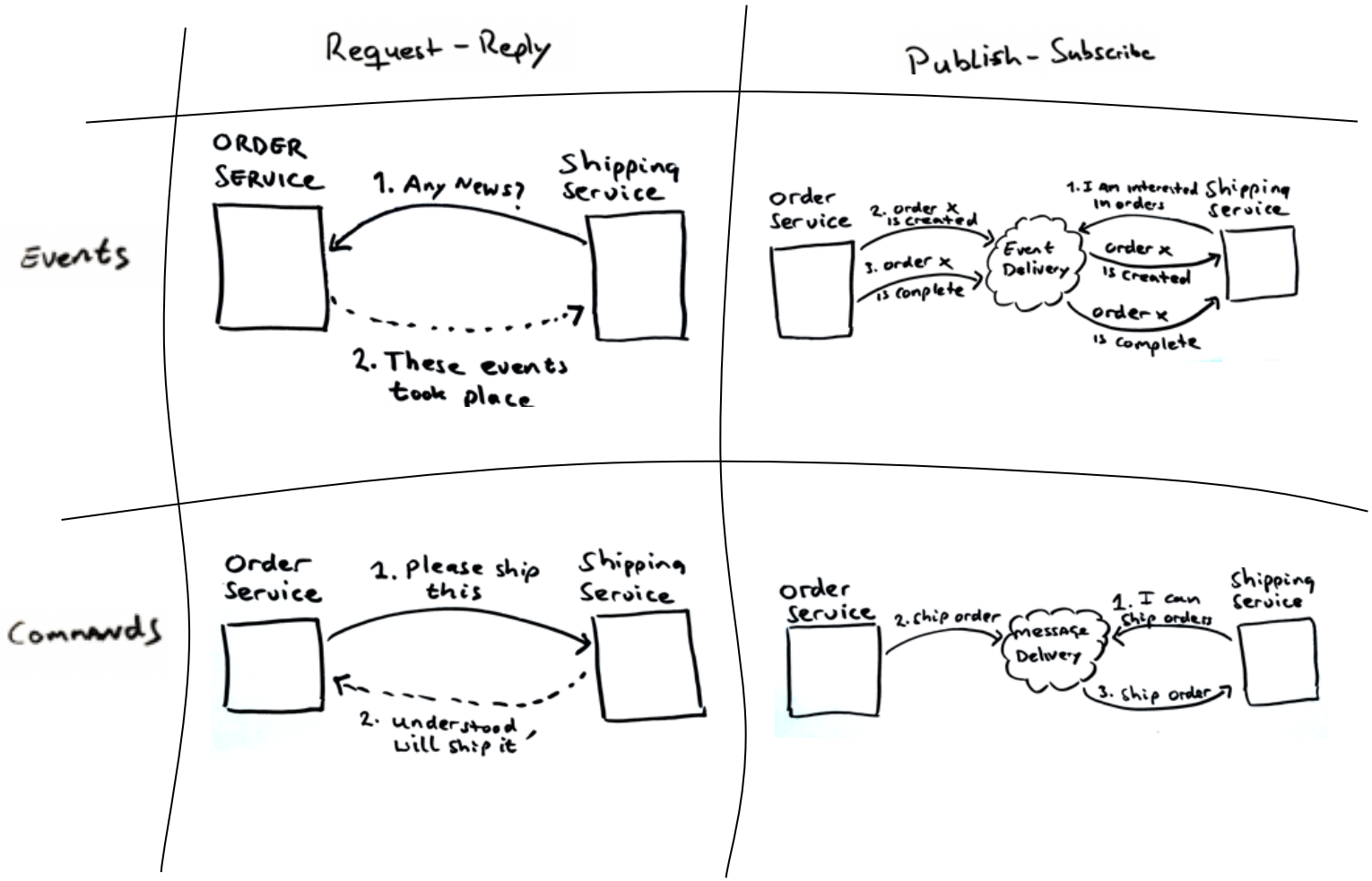
Event

https://msdn.microsoft.com/en-us/library/jj554200.aspx

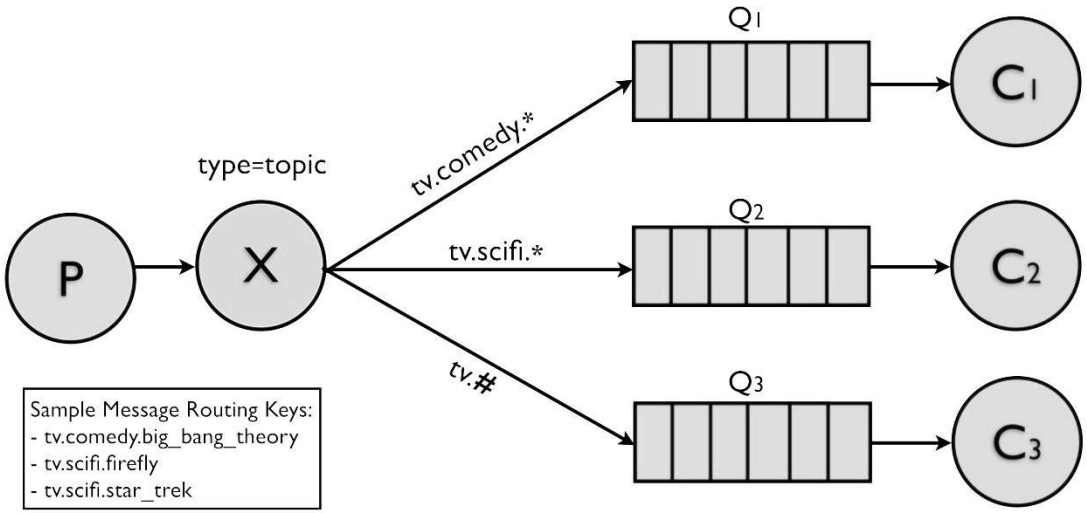
In computing, an event is an action or occurrence recognised by software that may be handled by the software. Computer events can be generated or triggered by the system, by the user or in other ways. Typically, events are handled synchronously with the program flow, that is, the software may have one or more dedicated places where events are handled, frequently an event loop. A source of events includes the user, who may interact with the software by way of, for example, keystrokes on the keyboard. Another source is a hardware device such as a timer. Software can also trigger its own set of events into the event loop, e.g. to communicate the completion of a task. Software that changes its behavior in response to events is said to be event-driven, often with the goal of being interactive.

https://en.wikipedia.org/wiki/Event_(computing)

Coupling Versus Autonomy



https://www.voxxed.com/blog/2015/04/coupling-versus-autonomy-in-microservices/

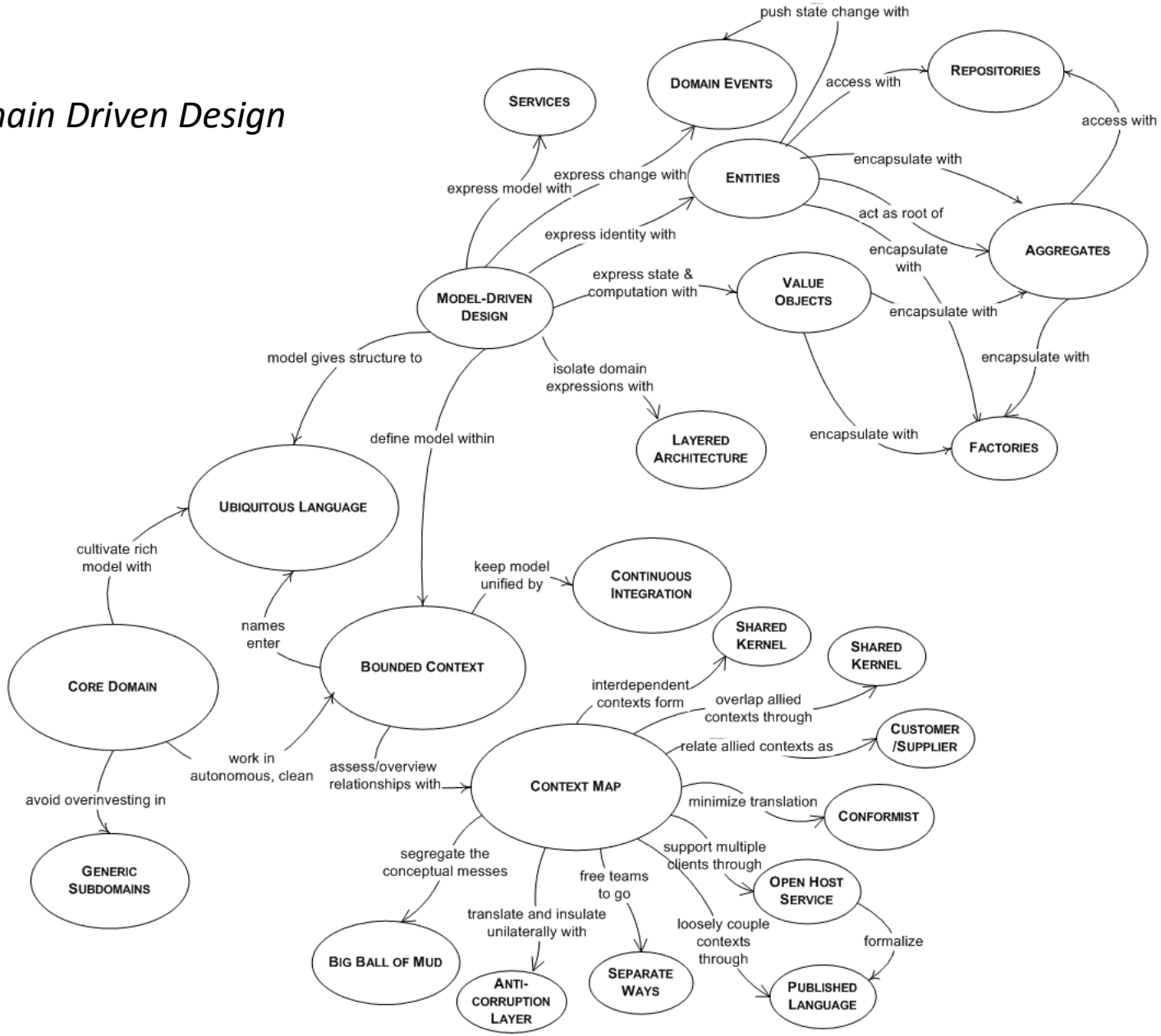


Queue

In computer science, a queue (/ˈkjuː/ kew) is a particular kind of abstract data type or collection in which the entities in the collection are kept in order and the principal (or only) operations on the collection are the addition of entities to the rear terminal position, known as enqueue, and removal of entities from the front terminal position, known as dequeue. This makes the queue a First-In-First-Out (FIFO) data structure. In a FIFO data structure, the first element added to the queue will be the first one to be removed. This is equivalent to the requirement that once a new element is added, all elements that were added before have to be removed before the new element can be removed. Often a peek or front operation is also entered, returning the value of the front element without dequeuing it. A queue is an example of a linear data structure, or more abstractly a sequential collection.

https://en.wikipedia.org/wiki/Queue_(abstract_data_type)

Domain Driven Design



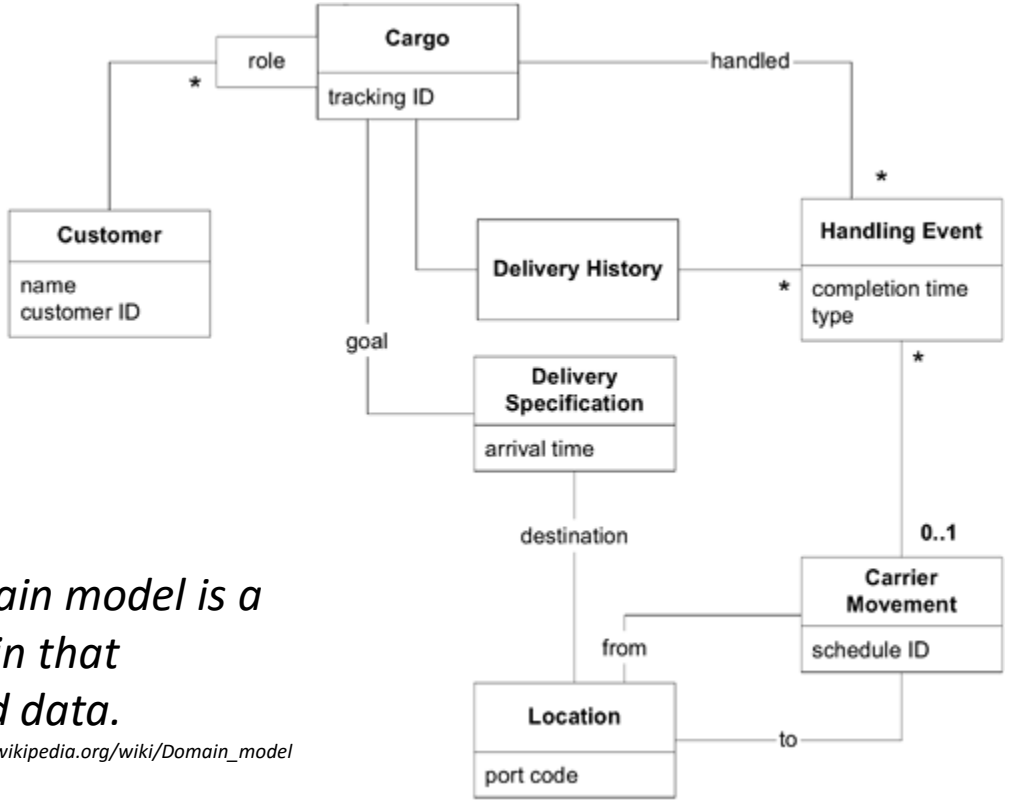
http://dddcommunity.org/book/evans_2003/

Ubiquitous Language

Programmers should speak the language of domain experts to avoid miscommunication, delays, and errors. To avoid mental translation between domain language and code, design your software to use the language of the domain. Reflect in code how users of the software think and speak about their work. One powerful approach is to create a domain model.

The ubiquitous language is a living language. Domain experts influence design and code and the issues discovered while coding influence domain experts. When you discover discrepancies, it's an opportunity for conversation and joint discovery. Then update your design to reflect the results.

http://www.jamesshore.com/Agile-Book/ubiquitous_language.html



Domain model

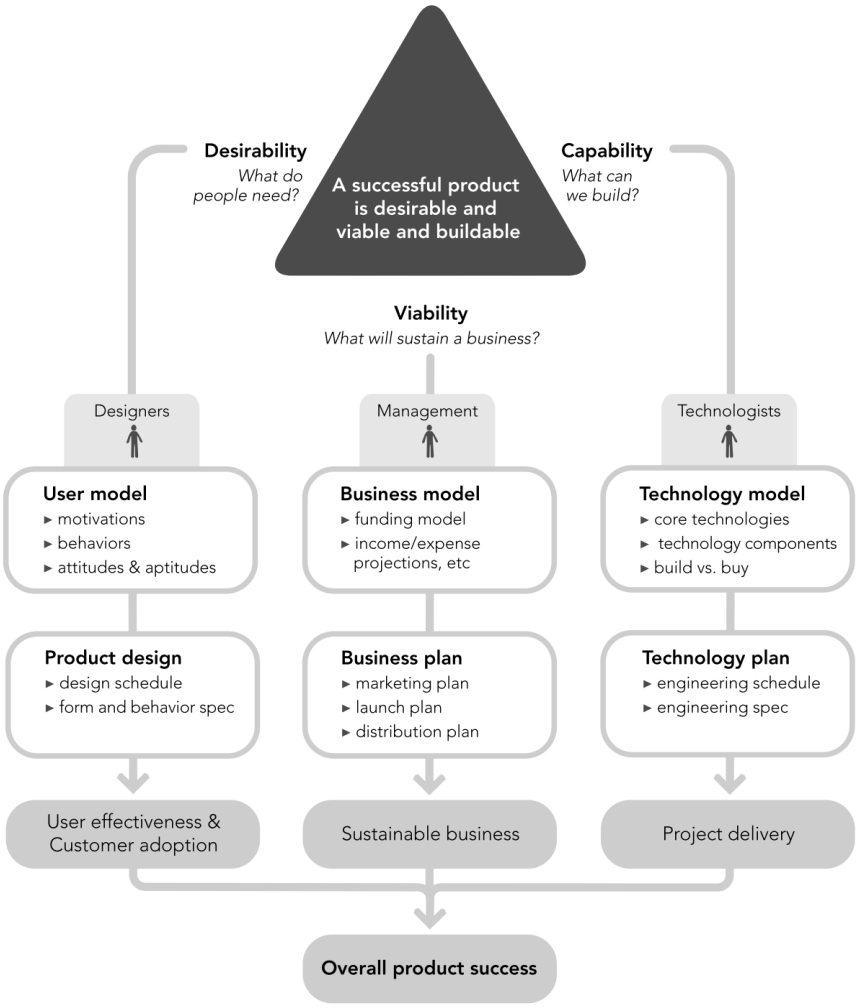
In software engineering a domain model is a conceptual model of the domain that incorporates both behavior and data.

https://en.wikipedia.org/wiki/Domain_model

Domain

A domain is a field of study that defines a set of common requirements, terminology, and functionality for any software program constructed to solve a problem in the area of computer programming, known as domain engineering.

https://en.wikipedia.org/wiki/Domain_(software_engineering)

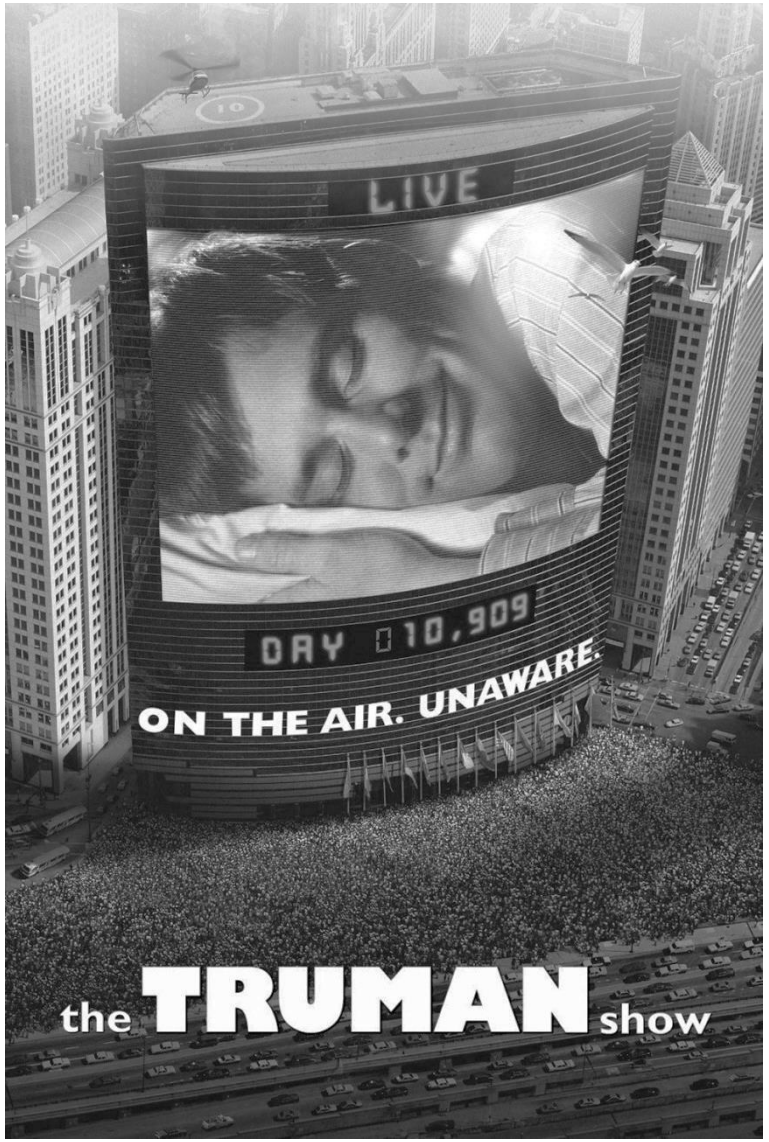


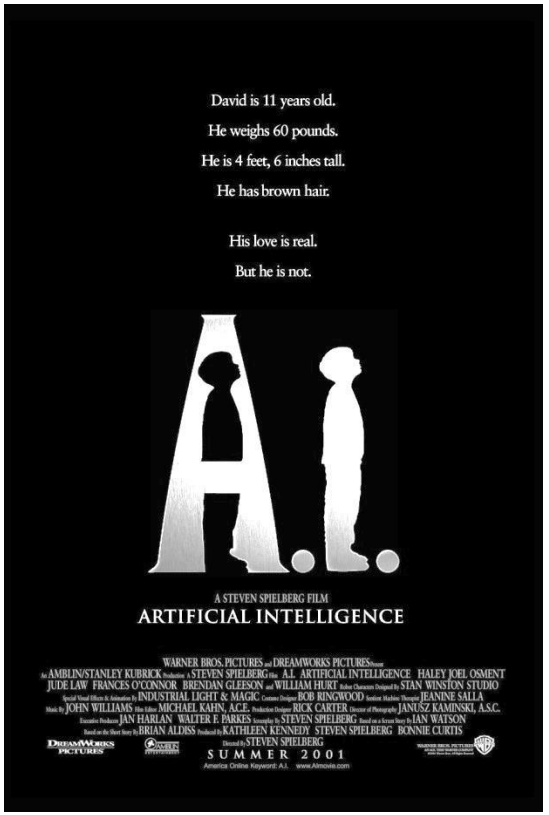
https://en.wikipedia.org/wiki/Alan_Cooper

Domain-specific language

A domain-specific language (DSL) is a computer language specialized to a particular application domain. This is in contrast to a general-purpose language (GPL), which is broadly applicable across domains, and lacks specialized features for a particular domain.

https://en.wikipedia.org/wiki/Domain-specific_language





Metaprogramming

Metaprogramming is the writing of computer programs with the ability to treat programs as their data. It means that a program could be designed to read, generate, analyse or transform other programs, and even modify itself while running. In some cases, this allows programmers to minimize the number of lines of code to express a solution (hence reducing development time), or it gives programs greater flexibility to efficiently handle new situations without recompilation.

https://en.wikipedia.org/wiki/Metaprogramming

“Talk is cheap. Show me the code.”