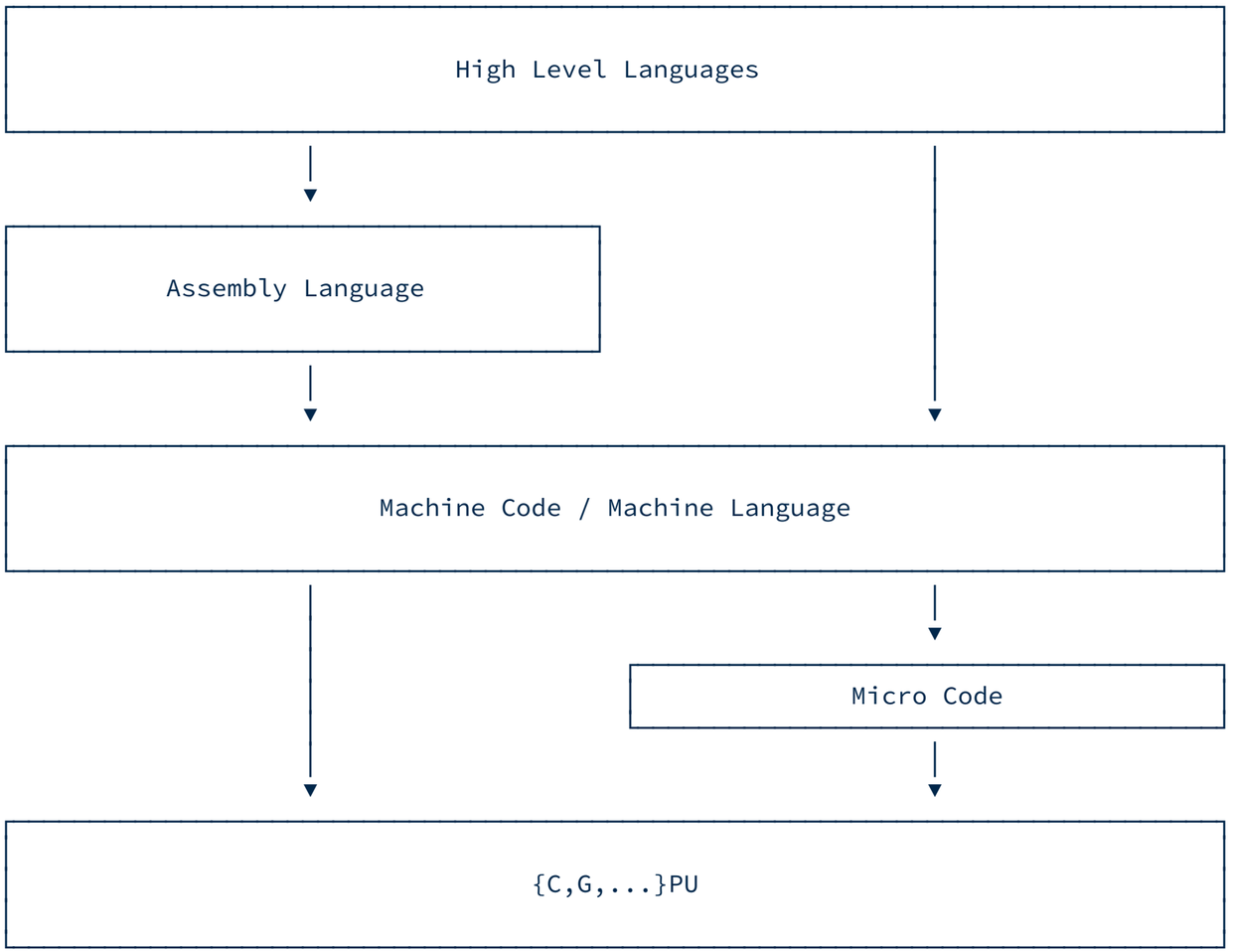
# Software

# What **is** Software?

Software is an *executable* process.

# Processes **Revisited**

# Software **Languages**



# High Level Languages

High level languages offer the *strongest* **abstraction** from the underlying hardware.

# Common Language Features

Expressions

Variables

Subroutines

Functions

Loops

Branching

Data Structures

Objects

Usability > Optimality

# *Compiled* vs. *Interpreted* Languages

**Compiled**

**Interpreted**

# Major High Level Languages

# **Many** Options

https://en.wikipedia.org/wiki/List*of*programming_languages

https://www.codingdojo.com/blog/7-most-in-demand-programming-languages-of-2018/

# Major Languages

| Language | Applications |
| --- | --- |
| Java | Android Apps, Major EHRs |
| Python | Cutting Edge Machine Learning |
| JavaScript | The Web, Servers |
| C/C++ | Embedded Systems, Highly Optimized Systems |

# **Programming** Tips

# **Embedded** Software

**98%** of all microprocessors are for *embedded systems*

— *Michael Barr* [1]

---

[1] Barr, Michael (1 August 2009). "Real men program in C". Embedded Systems Design. TechInsights (United Business Media). p. 2.

Consumer Electronics

Automotive

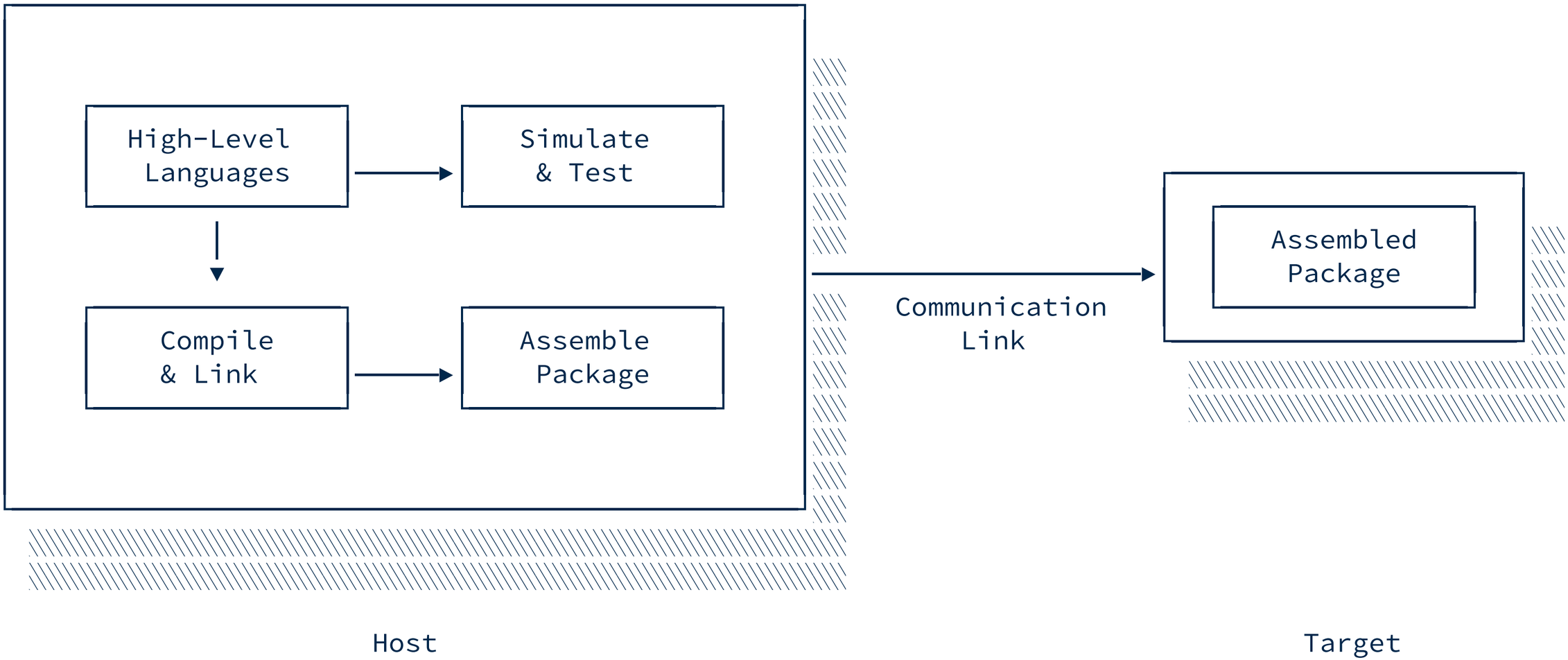
Medical Devices

Cooking Equipment

Industrial Systems

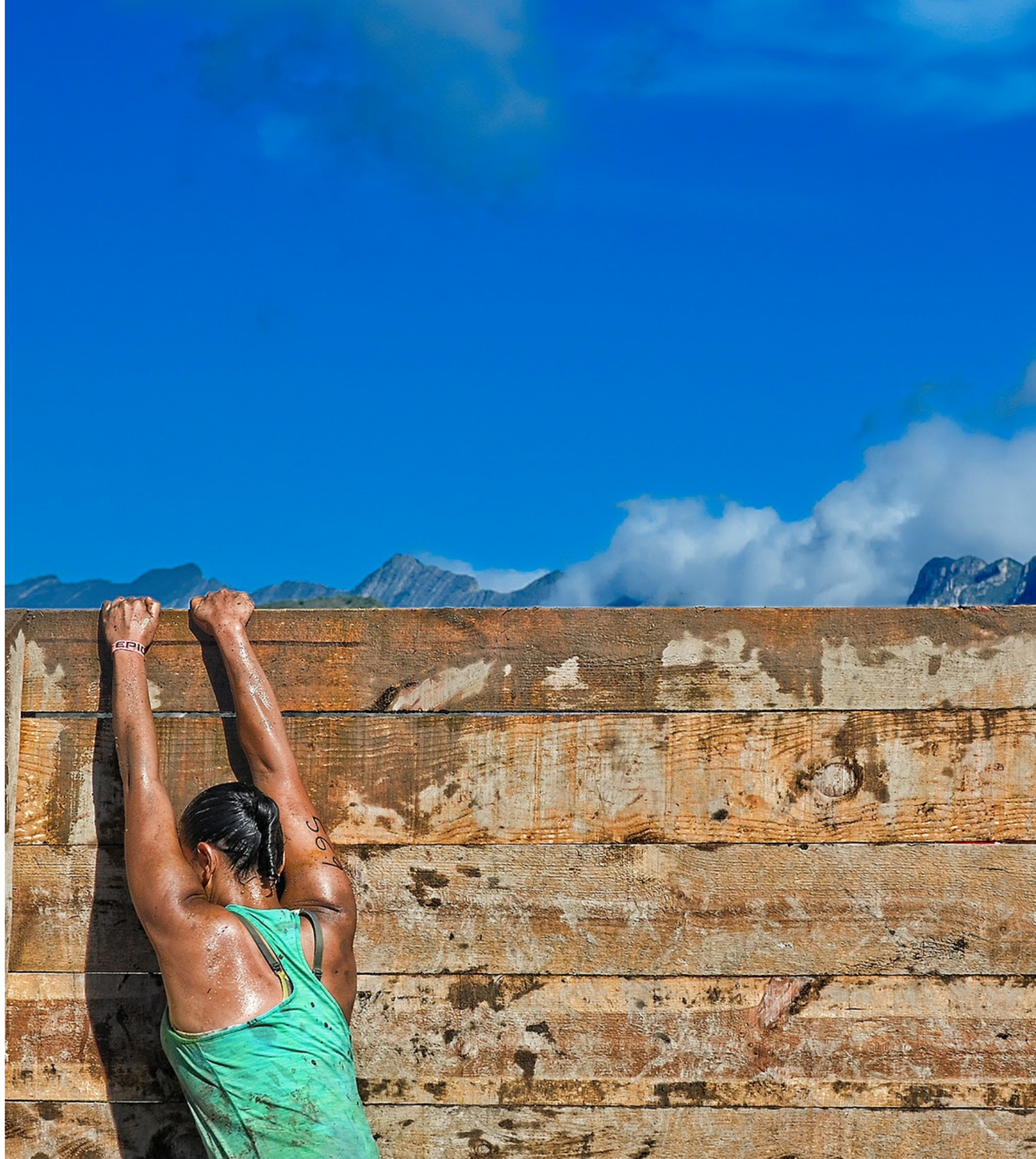
Military

# Embedded Software **Development**



# **Challenges** for Embedded Systems

- Limited resources
- Timing constraints
- Debugging without a user interface
- Robustness and longevity requirements



# Applications

An application is any program, or group of programs, that is designed for the **end user** … includ[ing] such things as *database programs*, *word processors*, *Web browsers* and *spreadsheets*.[2]

---

[2] Beal, V. (n.d.) Application (Application Software). In webopedia. Retrieved from https://www.webopedia.com/TERM/A/application.html

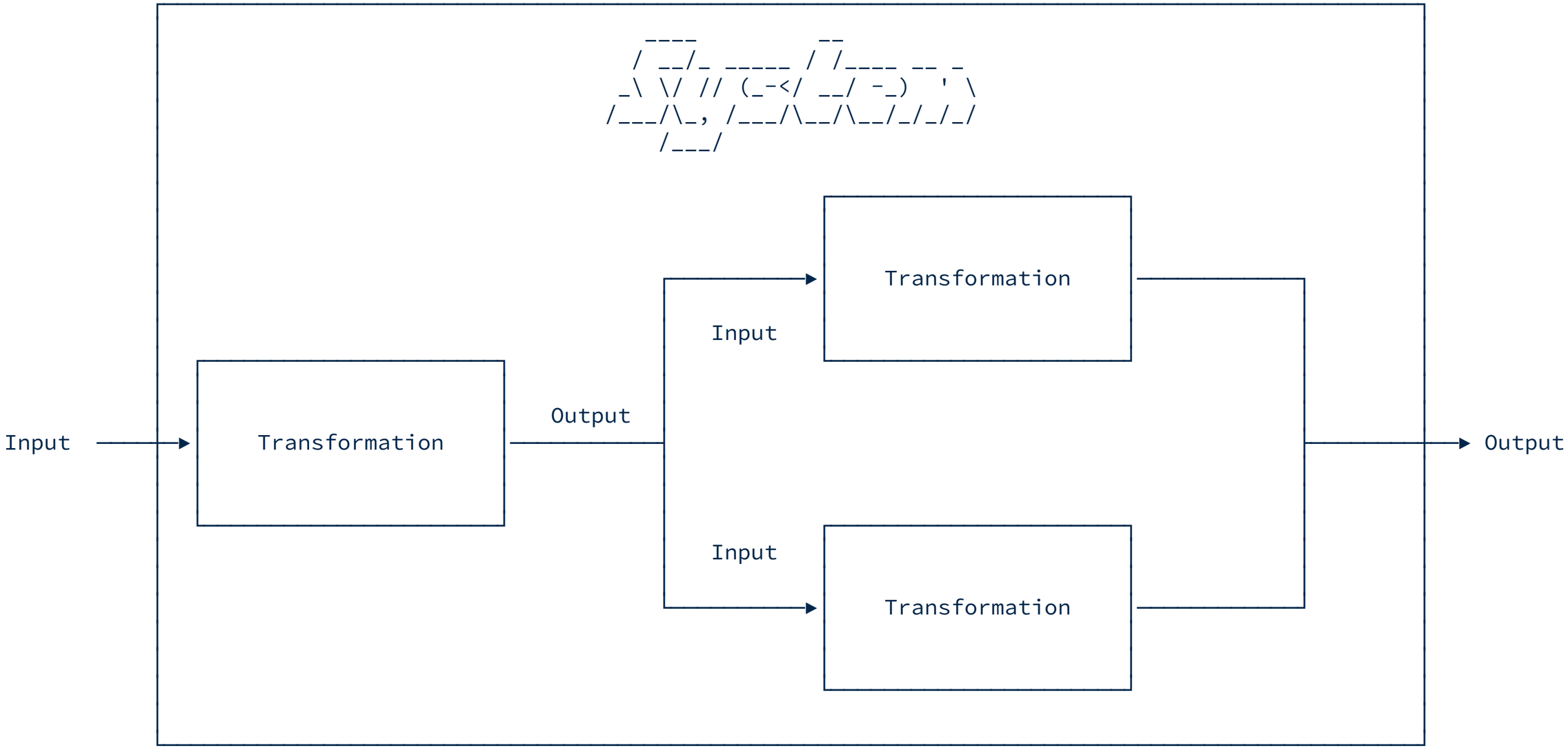
# **Apps** vs. **Applications**??

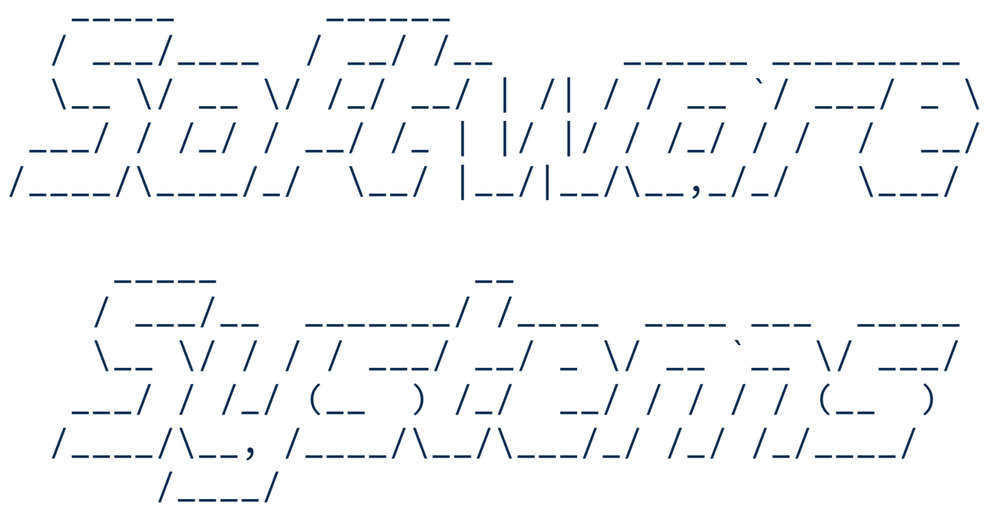
# Application **Interfaces**

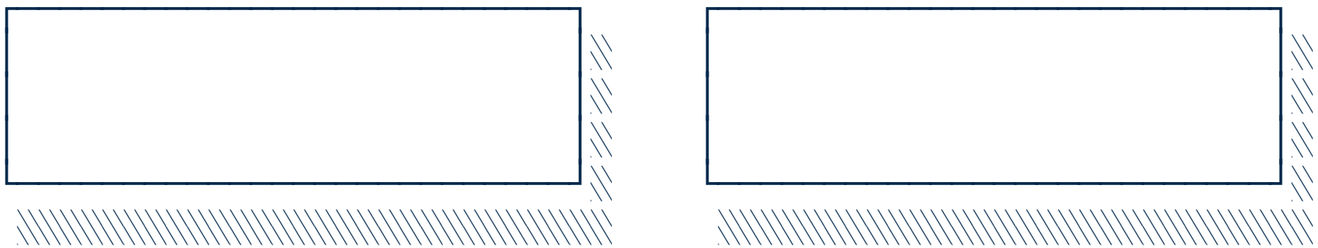
| **Command Line** | **Graphical** |
| --- | --- |
| vim | Notepad/TextEdit |
| ls | File Explorer/Finder |
| cURL | Chrome |

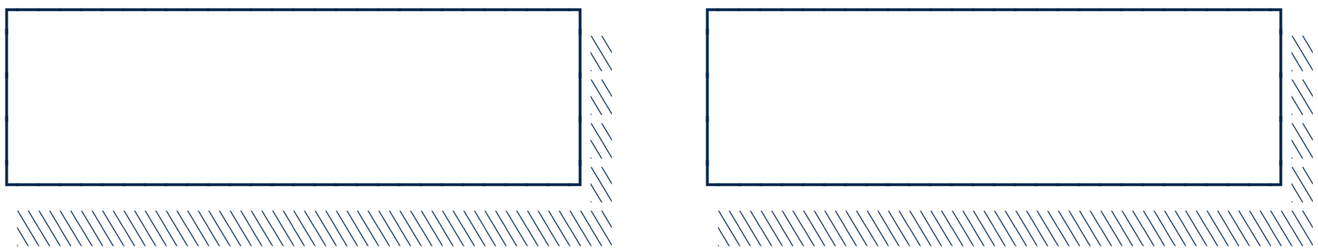
# Software **Systems**

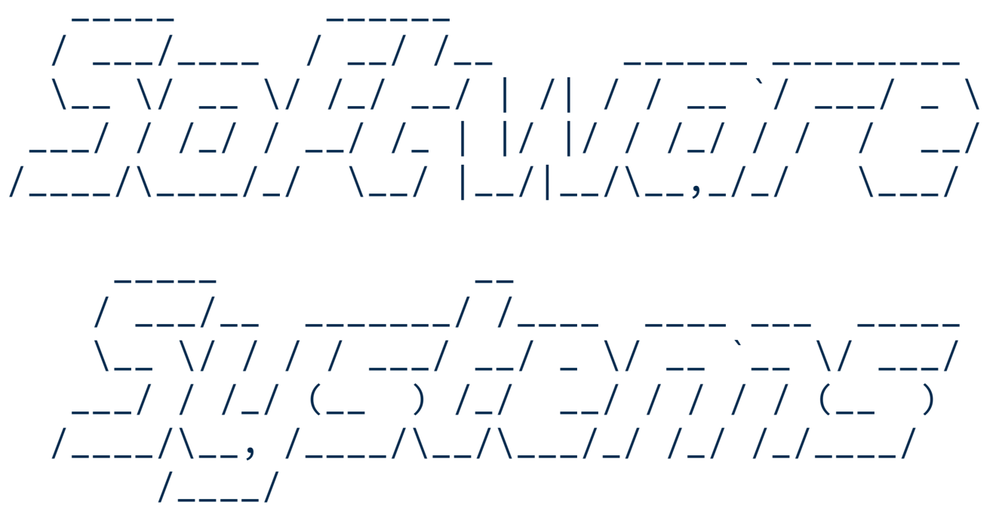
Process : System :: Software : Software System


____ __
/ __/_ _____ / /____ __ _
_\ \/ // (_-</ __/ -_) ' \
/___/_, /___/__/__/_/_/_/
/___/
Input
Transformation
Output
Input
Transformation
Input
Transformation
Output









Complex Applications

Numerous Users / Roles

Distributed / Networked

Integrated Functions

Software Systems are *designed* by Software **Engineers**

# Software **Architectures**

## Architectural styles and patterns [ edit ]

*Main article: Architectural pattern*

An architectural pattern is a general, reusable solution to a commonly occu
software design patterns.

Following traditional building architecture, a 'software architectural style' is a

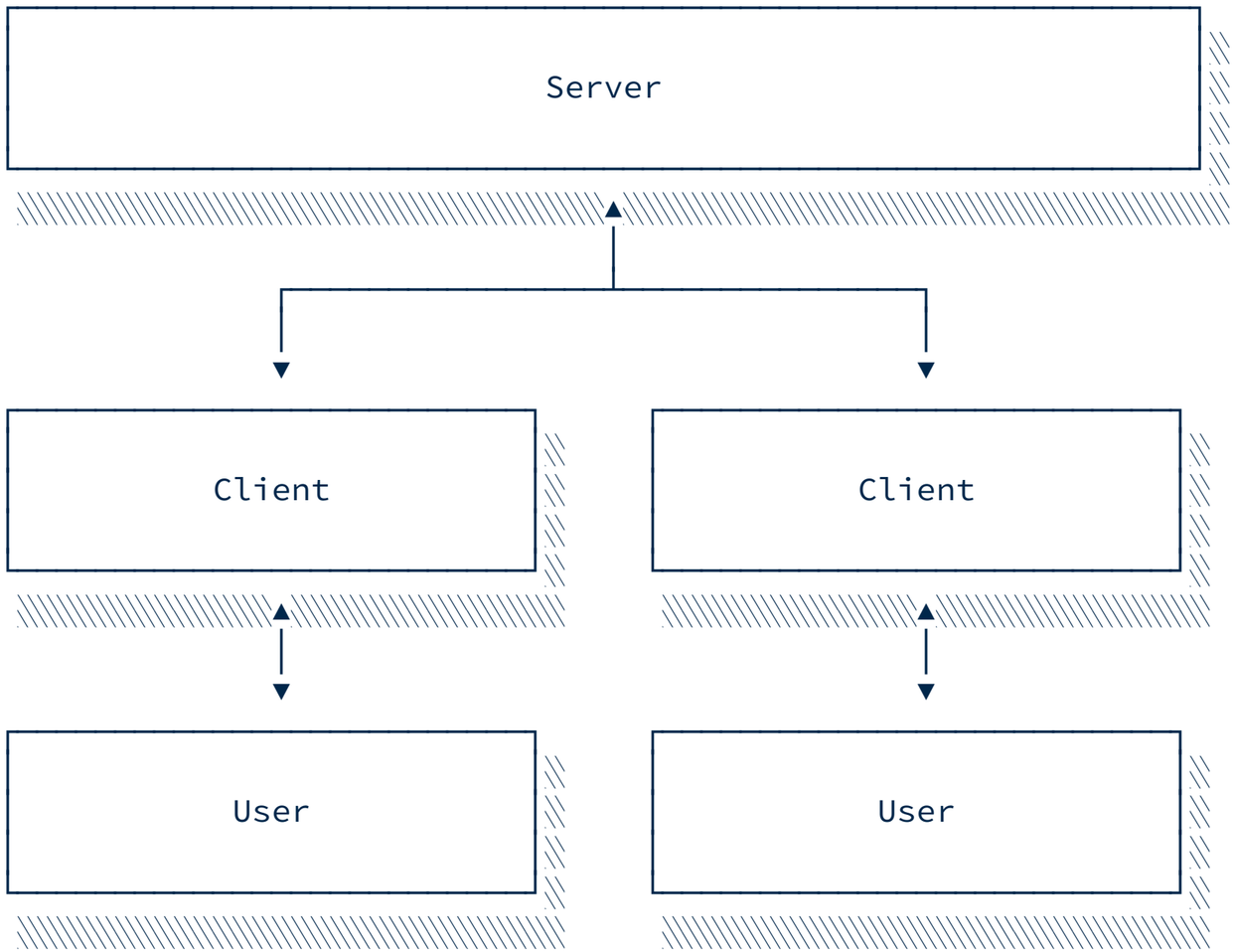
> *An architectural style defines: a family of systems in terms of a patt
> can be combined.*[33]
>
> > *Architectural styles are reusable 'packages' of design de*

There are many recognized architectural patterns and styles, among them:

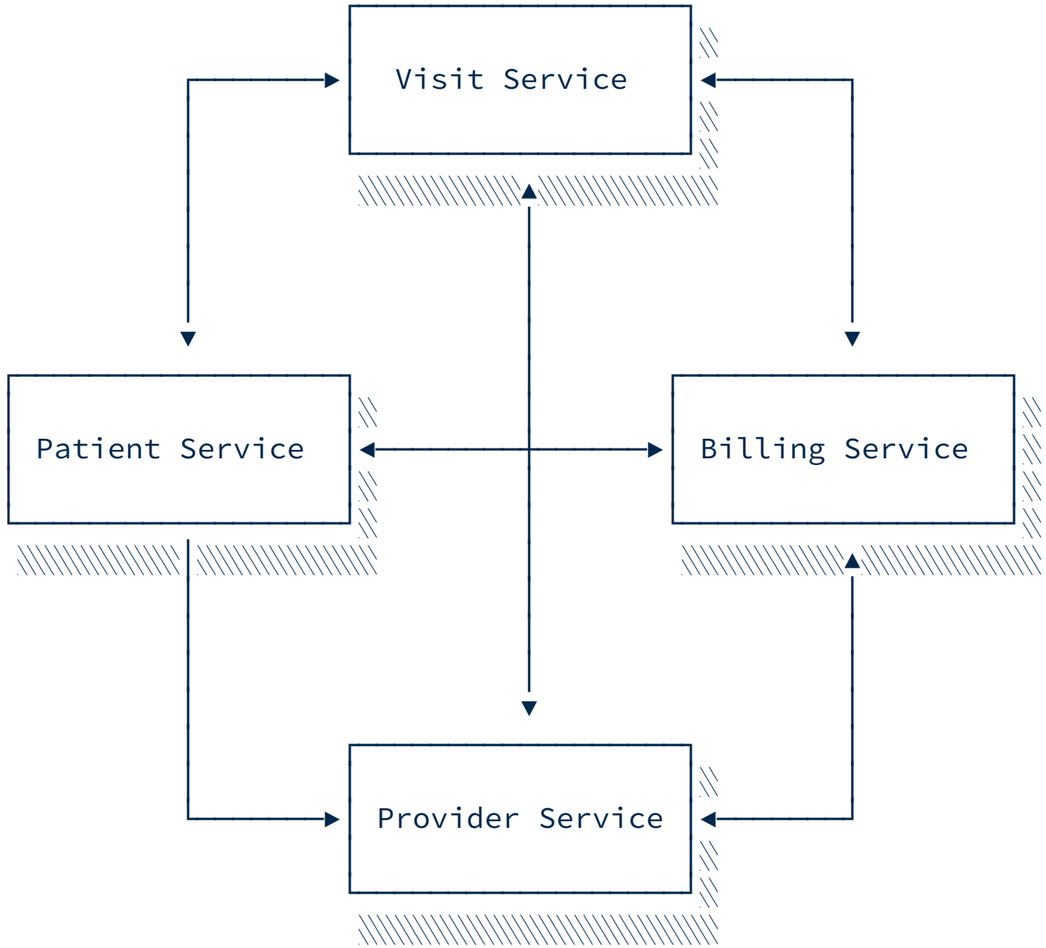
- Blackboard
- Client-server (2-tier, 3-tier, n-tier, cloud computing exhibit this style)
- Component-based
- Data-centric
- Event-driven (or implicit invocation)
- Layered (or multilayered architecture)
- Microservices architecture
- Monolithic application
- Peer-to-peer (P2P)
- Pipes and filters
- Plug-ins
- Representational state transfer (REST)
- Rule-based
- Service-oriented
- Shared nothing architecture
- Space-based architecture

Some treat architectural patterns and architectural styles as the same,[35] s
architects to use, they "provide a common language"[35] or "vocabulary"[33]

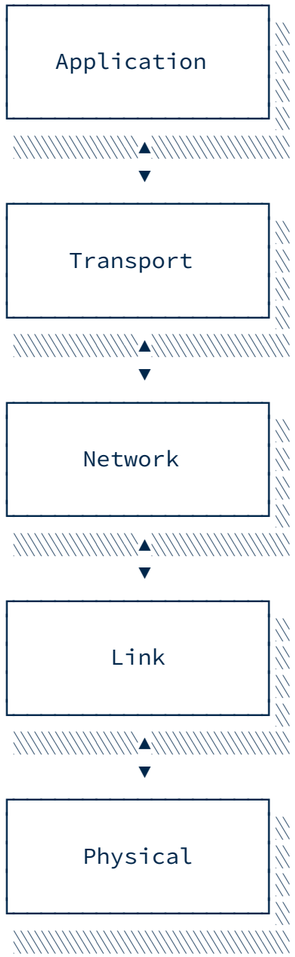
# Client - Server



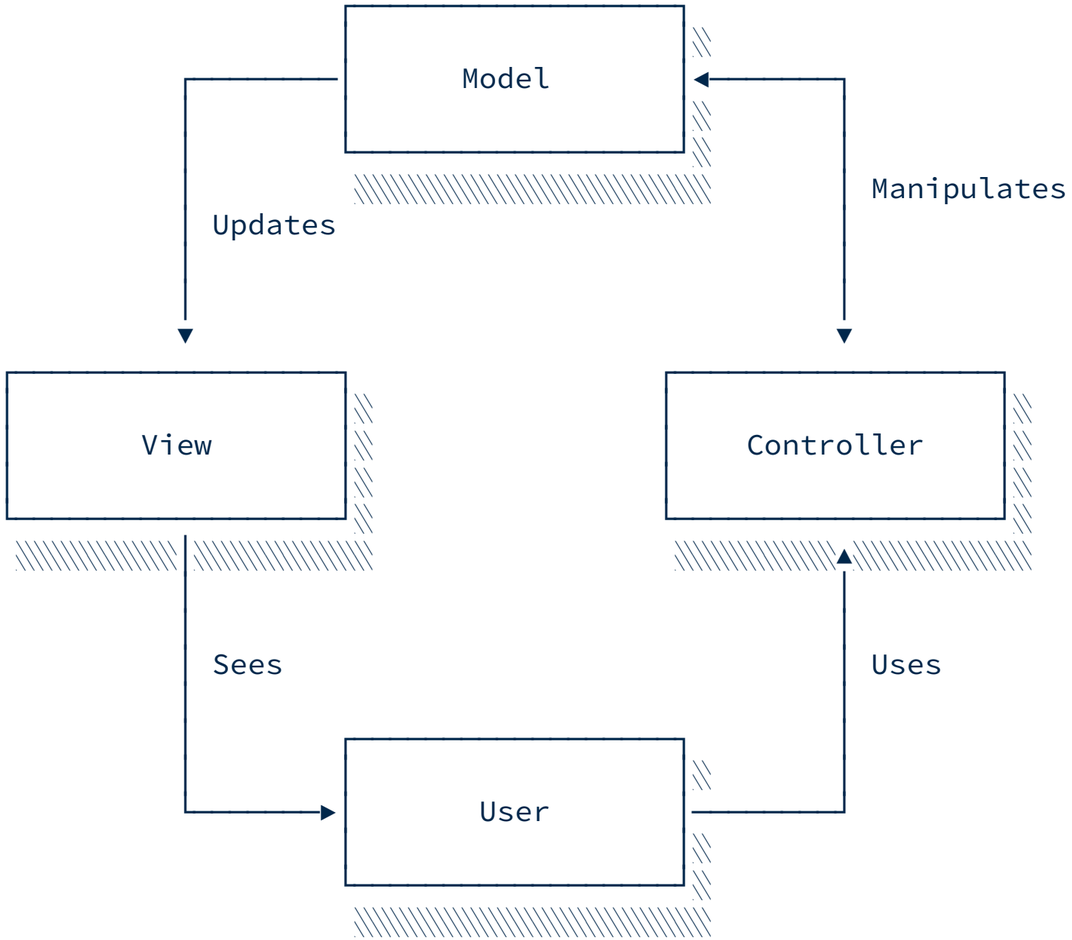
# Service-Oriented / Microservices



# Layered



# Model - View - Controller



# **Operating** Systems