



清华大学  
Tsinghua University

清华大学

计算思维八周课程

《第一周》

顾学雍

清华大学 iCenter

# 目录

---

● 计算思维

● 范畴论

● 逻辑模型

● Wiki和Git

● Markdown

---

# 01. 计算思维？

---

## 背景

全球化与人工智能的极速发展，加快暴露了国内教育体系的僵化现象，此现象普遍性地反映在师生的适应能力不良，从而阻碍了社会的正向发展。

## 目标

构建可以主动适应现代社会需求的学习场景，规模化地组织跨年龄、跨地域和超越专业界限的学习团队，通过参与者之间互通有无的真实学习需求，交互地引发学习本能，让学习参与者成为见证教学质量的利益相关者，并根据学习参与者的过程数据总结出来的科学结果，客观地优化整个民族的学习策略。

### 效果

- 以现有的网络化教学资源与通信基础设施建设，规模化地调动学习者互为教和学的辅助对象，根本上重构全球学习的生态结构。
- 从以学校为单位的人力教学成果见证机制，转换为全网范围的公开能力见证机制。
- 利用通信与机器学习的技术，用最快的速度，把最相关的知识内容，经由一个可以因时、因地制宜的学习组织方式，传播给有需求的学习者。

### 输出

- 运营机制：一个网络化的开源教学任务协同运营工作标准，和根据该标准配套的运营团队，调度实体校园和众创空间的资源，组织全球化的教学活动。
- 动态学历：使用区块链等相关分布式数据见证技术，为个人，团队，社群学习单位，提供可相互见证的过程数据。
- 学习资源搜索引擎：将网络化的知识内容、可提供教学服务的专家、器材与场地资源，以及有意愿参与学习的个人与团队，整合为可被动态搜寻的数据内容，成为透明共享的资源。

### 过程

- 多尺度的教学实验：从小型工作坊、全学期的课程，到专门定制的学位课程体系，乃至跨院校与产业结合的教学协同活动等等，持续验证与磨合教学过程和相关的运行团队。
- 根据多种课程内容，归纳出一套可通用于不同教学活动的过程数据的分类方式与相应的统计分析方法和信息容器，用于提供科学化和个人和团队的学习成效反馈。
- 开发一套可部署于个人计算机和数据中心的知识容器标准，与可针对此容器内容格式做数据挖掘和搜索的软件系统。

### 输入

- 超越学科界限的知识组织方法论：“超限学习过程”。
- 超限学习开发团队：
  - 知识容器的开发团队
  - 教学实验运行团队
  - 教学工具开发团队
  - 行政工作执行团队
- 运营资源：
  - 5年x每年2亿人民币预算
  - 国内外教育系统的联系机构：(UNESCO/OECD)
  - 国内外学校与创客空间的实体教学环境。

## 外部因素

在本项目尚未建立成熟的运营机制、包括有效的动态学历的网络化见证体系之前，前期参与筹划与执行的核心人员配置和权力分配结构，对项目的执行方向与发展速度会有决定性的影响。

## 背景

计算思维是贯穿整个课程的轴心

## 目标

让学生了解计算思维是甚么  
让学生了解学习计算思维的作用

### 效果

根据学生提交的逻辑模型

评估学生对逻辑思维的接受程度: 是否表现出能将理论应用于实际问题的转换能力

评估学生对逻辑模型的理解

### 输出

学生需要挑选一种逻辑思维并填写其逻辑模型 (待议) 自己搜索逻辑思维相关素材利用逻辑模型描述事件

### 过程

讲述计算思维的起源  
讲述计算思维是甚么  
讲述计算思维的作用

### 输入

2006年发表于《Communications of the ACM》杂志上的《Computational Thinking》的相关翻释材料

2012年10月在天津召开的微软亚太教育峰会上的访问记录

## 外部因素

# 引言

“这一切都在为一场变革而准备。我们就像青蛙待在呆在不断加热的水当中，它最终会沸腾。我们必须决定是在升温的水中继续游动，还是开始作出其他抉择。”

# 事件史 计算思维课程起源



Noam Nisan, Professor  
Computer Science and Engineering

## Nand2Tetris

“Together with Noam Nisan, I developed [Nand2Tetris](#) – an open-source approach for teaching applied computer science, described in a bestselling [MIT Press book](#) and in a 2012 [TED talk](#). This project evolved into one of the first successful MOOCs (Massive Open Online Courses), taken freely by thousands of self-learners over the web since 2005. I also teach this course on [Coursera](#), together with Noam.”

--Shimon's Blog



Shimon Schocken, Professor  
Computer Science

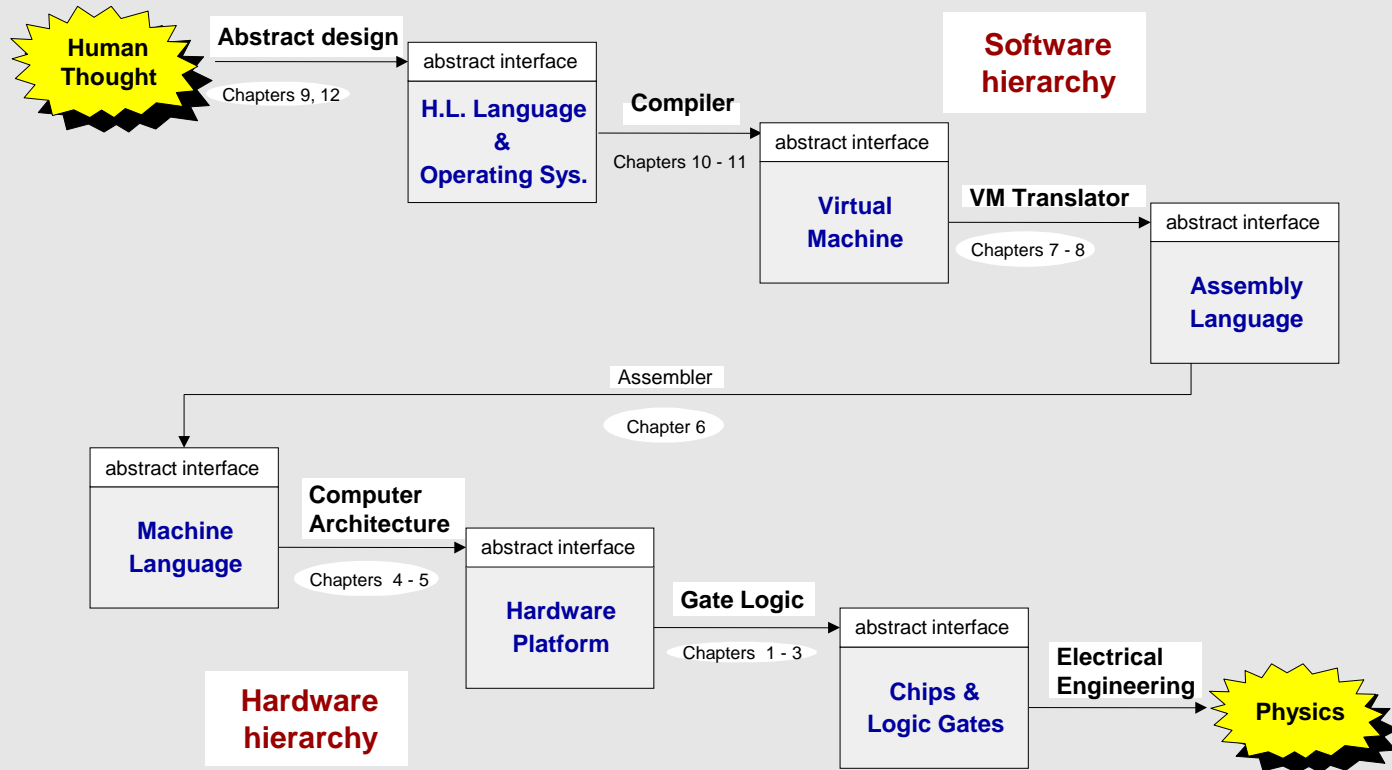
“这门课程曾经是哈佛大学计算机系的一门公共课，由于其内容实在是太强大，被众学生戏称为CS101。”

--豆瓣读者[Kayuk](#)评价

<https://www.coursera.org/learn/build-a-computer>

<https://book.douban.com/review/7115224/>

# The Big Picture 原始的12周Nand2Tetris课程内容与结构



(Abstraction–implementation paradigm)



Read the documents on [nand2tetris.org/papers.php](http://nand2tetris.org/papers.php)

## A Synthesis Course in Hardware Architecture, Compilers and Software Engineering

Shimon Schocken  
IDC Herzliya

Noam Nisan  
Hebrew University

Michal Armoni  
Weizmann Institute

March, 2009

### ABSTRACT

We describe a synthesis course that provides a hands-on approach to hardware and software topics learned in computer science. Over twelve projects, we walk the students through the gradual construction of a platform and a modern software hierarchy, yielding a basic understanding of the process of building the computer, the students gain a first-hand experience of how software systems are designed and how they work together. This document contains all the materials necessary to run this course in operation.

## Taming Complexity in Large-Scale System Projects

Shimon Schocken

Efi Arazi School of Computer Science

IDC Herzliya

September, 2011

### ABSTRACT

Engaging students in large system development projects is an important educational objective, since it exposes design and programming challenges that come to play only with scale. Alas, large scale system projects can be monstrously complex – to the extent of being infeasible in academic settings. We describe a set of principles and a framework that enable students to develop large-scale systems, e.g. a complete hardware platform or a compiler, in several semester weeks.

# 模块化项目 + 自动检验工具

Hardware Simulator (2.5) - /Users/bkoo/Documents/Workspace/GitRepos/nand2tetris/02/ALU.hdl

File View Run Help

Chip Name: ALU Time: 0

Input pins

Name	Value
x[16]	0
y[16]	0
zx	0
nx	0
zy	0
ny	0
f	0
no	0

Output pins

Name	Value
out[16]	0
zr	1
ng	0

HDL

```
// This file is part of www.nand2tetris.org
// and the book "The Elements of Computing Systems"
// by Nisan and Schocken, MIT Press
// File name: projects/02/ALU.hdl

/*
 * The ALU (Arithmetic Logic Unit)
 * Computes one of the following:
 * * x+y, x-y, y-x, 0, 1, -1, x, y
 * * x+1, y+1, x-1, y-1, x&y, x|y
 * according to 6 input bits
 * In addition, the ALU computes
 * * if the ALU output == 0, zr is 1
 * * if the ALU output < 0, ng is 1
 */
```

Internal pins

Name	Value
zxOut[16]	0
nxOut[16]	-1
zyOut[16]	0
nyOut[16]	0
yOut[16]	0
xPlusY[16]	0
xAndY[16]	0
fOut[16]	0
nOut[16]	-1
lowerBits[8]	0
higherBits[8]	0
zrl	0

File View Run Help

Program

Running...

Argument

This

That

Temp

Global Stack

Address	Value
256	0
257	0
258	0
259	0
260	0
261	0
262	0
263	0
264	0
265	0
266	0
267	0
268	0
269	0
270	0

RAM

Address	Value
SP	0
LCL	1
ARG	2
THIS	3
THAT	4
Temp0	5
Temp1	6
Temp2	7
Temp3	8
Temp4	9
Temp5	10
Temp6	11
Temp7	12
R13	13
R14	14

CPU Emulator (2.5) - /Users/bkoo/Documents/Workspace/GitRepos/nand2tetris/06/Pong.asm

File View Run Help

ROM

Address	Value
0	0x256
1	D=A
2	@0
3	M=D
4	@133
5	0;JMP
6	@15
7	M=D
8	@0
9	AM=M-1
10	D=M
11	A=A-1
12	D=M-D
13	M=0
14	@19
15	D;JNE
16	@0
17	A=M-1
18	M=-1
19	@15
20	A=M
21	0;JMP
22	@15
23	M=D
24	@0
25	AM=M-1
26	D=M
27	A=A-1
28	D=M-D

RAM

Address	Value
0	321
1	315
2	308
3	4880
4	2869
5	352
6	0
7	0
8	0
9	0
10	0
11	0
12	0
13	6959
14	322
15	7188
16	4872
17	2850
18	2868
19	0
20	16384
21	-1
22	7780
23	4
24	2185
25	2118
26	3495
27	16384
28	-1

Game Over

Score: 1

D: 0

ALU

D Input: 0

M/A Input: 0

ALU output: 0

# 起源

2006年3月，美国卡内基·梅隆大学计算机科学系主任，周以真教授，在美国计算机权威期刊《Communications of the ACM》杂志上发表

《计算思维》(Computational Thinking)

背景

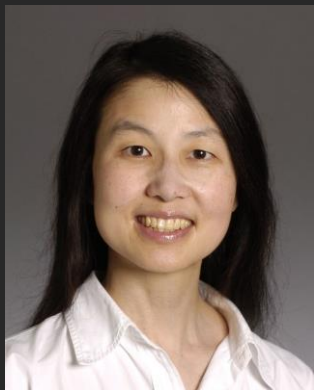
03、04年间在美国选修计算机科学本科生越来越少，情况让人沮丧和担忧  
其实在大数据时代甚至互联网时代以前，计算机科学的发展并不如现在的精彩  
计算机为其他科学(如物理和化学)研究服务，并没有作为一个核心技术被重视  
第二次人工智能浪潮于20世纪80年代开始，在2000年前后破灭  
现象也就是导致当时选修计算机科学本科生少的原因

随后

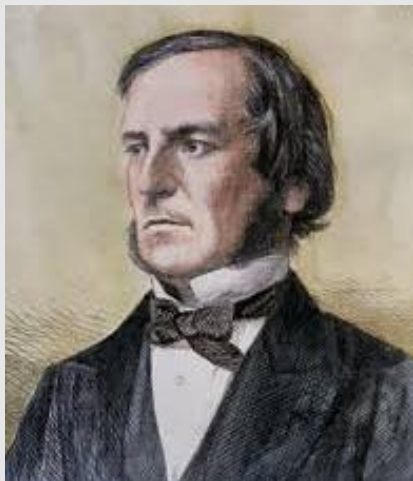
互联网时代兴起，计算机网络技术受到关注  
信息时代爆发，大数据计算技术被提升到一个战略层次的技术  
计算机科学实实在在地影响著其他各个专业领域时

此时

我们回归根本，思考计算机科学背后的思维是甚么  
既然计算科学很大程度地影响了其他的专业领域，那计算思维是否也能被迁移到其他场景？



# ► Three Giants in the History of Computation



**George Boole**  
**Boolean Algebra**



**Claude Shannon**  
*A Symbolic Analysis of Relay  
and Switching Circuits*



**Bertrand Meyer**  
**Design by Contract**

# 计算机科学

- 计算机的整体能力在不断加强

能力越大，责任越大

- 随著计算机性能提高，问题也会变得复杂
- 计算能力，存储能力，通信速度始终不能完全满足当下的需求



# 计算机科学

- 计算机科学要解答的是: 对某个特定的问题, 在现有计算机能力的限制下
  - 计算机**能不能**解决?
    - [例] 判断一个股票我现在该不该买
  - 如何利用计算机**来**解决?
    - 怎么以计算的方法解决问题?
    - [例] 用人工神经网络对数值进行预测
  - 如何利用计算机**更好**地解决问题?
    - 哪个方法更优甚至最优
    - [例] 目测新的人工神经网络框架不断被提出, 用最新的框架来尝试
  - 如何利用计算机**充分好**地解决问题?
    - 近似解是否就足够?
    - 是否允许误差?
    - [例] 或许不我 need 知道股票具体涨多少, 我只需要知道它涨还是不涨

# 计算机科学

- 在解答这些问题的过程中，计算机科学家本质上做了以下的步骤
  - 先对这个问题进行抽象
    - [例] 股票可以用一套数字的指标来描述，不需要知道它具体的背景
    - 排除一些无关的因素
  - 对它重新以可计算的方法表达
    - [例] 一系列数字随著时间变化，我们需要预深它下一刻会怎么变
    - 当问题被描述为可以被计算的，那么我们只需要进行数字的计算，计算的东西本身是否股票并没有区别
  - 考虑这个问题的表达是不是准确
    - [例] 抽象以后是否把关键的指标省略了
    - 确保方式抽象的过程无误，这样计算的结果才能对应到原始问题的解
  - 考虑这个问题的解决效率是不是高
    - [例] 选择的人工神经网络框架计算得快不快，会不会预深 1 个小时后的变化需要 2 个小时的计算

# 计算思维是什么？

- 计算思维受启发于计算机科学

- 计算机科学中有很多方法论
- [例] 计算机软硬件架构中，**虚拟机**引出虚拟存储空间的概念
- [例] 计算机网络结构中，**多层结构模型**引出多层次设计的概念
- [例] 大数据计算中，**Map-Reduce**引出了大任务拆成可以同时进行的小任务再汇总的概念
- [例] 软件系统设计中强调将一个大型系统拆解成**低耦合的模块**
- 周以真提出的正是从这些计算机科学中的技巧**提取出抽象的方法论**，并应用于**解决其他问题**
  - [注] 从具体技巧中提取出抽象的方法论本身也是一种计算思维的体现，可以说计算思维孕育了计算思维本身



# 计算思维是什么？

- 换言之，计算思维

- 是解决问题的一种切入角度

- 强调将一个问题清晰、抽象地描述出来，并尝试以计算的方法解决

- 提高解决问题的效率

- 计算思维已经将计算机科学中的技巧归纳有限几个能解决目前大部分问题的方案
    - 不管是日常生活里的问题，商业问题，创新发展等等

- "计算思维是每个人的基本技能，不仅仅属于计算机科学家，在阅读、写作和算术（英文简称3R）之外，我们应当将计算思维加到每个孩子的解析能力之中"

# 计算思维的特点

- 唯一性

- [例] 在代数计算中,  $1+1$  必然是2; 在自然语言中, 却有形如" $1+1$ 大于2", " $1+1$ 大于3"的说法
- [例] 在计算机硬件架构中, 底层是物理电路, 相同的电压输入必然会给出相同的电压输出 (除非电路受损)
- [例] 非专业的人会很恐惧使用操作系统中黑底白字的终端, 但专业人员会知道拷贝相同的指令就会有相同的结果 (除非拷漏一个字符)

- 严谨性

- [例] 业内经常听到某公司新员工把数据库给删了, 把电脑根目录给删了, 计算机会严格完成指令, 而并不会考虑你是不是故意删除
- [例] 相对地只要指令没有输入错, 电脑不会无故把文件删除

# 计算思维适合什么人学习

计算机专业人员



但学过计算思维的计算机专业人员反映学习计算思维依然有相当帮助

- 重新理解他们正在使用的技术背后的原理
- 更好地利用现有的工具

- 更容易理解和上手新的技术

计算机科学爱好者



计算思维抽象地说明了计算机科学的原理  
学习计算思维更有利于学习计算机科学

非专业人士



计算思维的泛用性很广，它正改变着各个学科的思考方式

- 生物学里的计算生物学
- 经济学里的计算博弈理论
- 物理学家里采用的量子计算
- 化学里采用纳米计算

# 学习目标

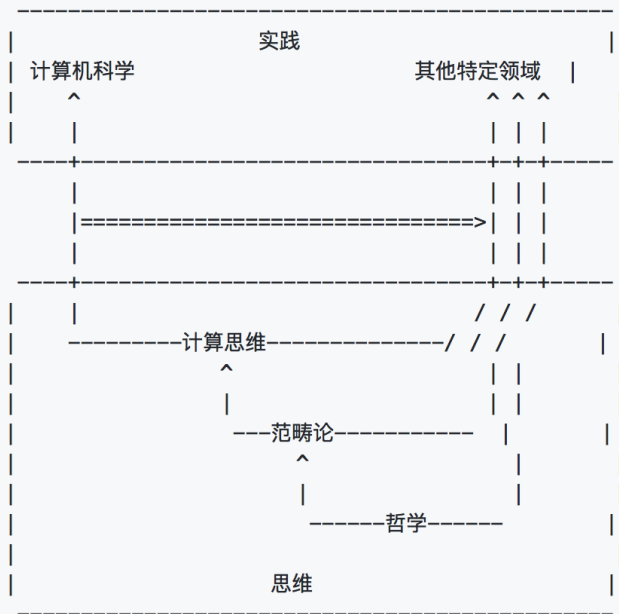
- 习惯利用计算解决问题

- 意识到当代计算能力的强大，就知道比如说大规模的或者复杂的问题可以提用一些简单的计算技巧去解决
- 现在大数据在任何领域都很火
  - 像生物、医药、金融、社科、人文、还有基础科学方面，他们每个领域都有很多的大数据
  - 通过计算技术, 运用超算的基础设施, 可以从大数据中去挖掘一些内容, 解决这个行业的前沿问题

- 懂得将计算机科学中的方法论应用于其他专业领域问题，甚至日常

- 即使不是以计算思维的名义出现，计算思维在日常生活中无处不在
  - [例] 把一个教室的椅子搬到另一个教室 - 并行计算
  - [例] 一个工作团队按职能分工 - 模块化
- 但我们强调的是当遇到新的问题，考虑用这些现有的方法去解决

# 为什么是“计算”思维



- 和文科相比更具体更严谨
- 计算机科学在广度发展比其他理工科更广
- 计算机科学比数学更直接地解决实际问题

---

## 02. 范畴论

---

## 背景

逻辑模型是本课程的重要工具之一  
本课程的很多内容都以范畴论来解释范畴论是一种强大的逻辑演算工具

## 目标

让学生认识范畴论的基础概念

### 效果

根据学生提交的报告

评估学生对  
范畴论的理解  
评估学生对  
范畴论的接受  
程度

### 输出

找出一种日常生活中  
可以用范畴论表示的  
事例，撰写对应报告  
或针对范畴论的理论  
撰写自己的感想或若  
无法理解范畴论的作  
用，解释自己的想法

### 过程

发展历史与概述  
范畴论核心概念  
使用范畴论的作用  
范畴论的应用案例

### 输入

Conceptual  
Mathematics

## 外部因素

# 发展历史与概述

范畴论的核心概念在1945年由Samuel Eilenberg和Saunders Mac Lane从拓扑学引进而来

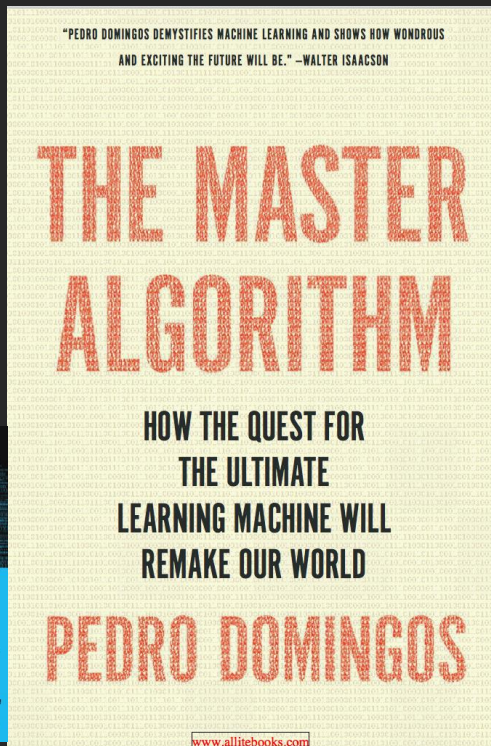
- 目的在于理解自然转换
- 为此，必须定义函子
- 为了定义函子，就自然地要引进范畴

范畴论是抽象地处理数学结构以及结构之间联系的一门数学理论

被开玩笑地称为“一般化的抽象废话”



# Will “The Master Algorithm” Reframe Our World?



Bill Gates' recommendation

***The main idea of this book:***

*Processing data through a single universal learning algorithm— can get all the knowledge from the past, the present, and the future*

*Pedro Domingos, TMA, P. 25*



# 教育部资助的清华大学学校特色项目： ABC 认知科学与计算思维的课程体系：

## 任务 执行方

刚入门的新生导引课程，  
全面体验数字社会的群体  
运作基本机制



A. Application Orientation (宏观决策者)

Strategy Analysis based-on Computational Thinking

Industry Frontier

## 挑战 设计方

自愿参与课程设计的  
前期学生

B. Business Operations (技术架构师)

基于数字化工作流的软件与硬件架构设计师

基于计算思维的系统设计

C. Cybernetics Infrastructures (各专业领域的从业人员)

为各领域定制的虚拟 /  
增强现实

知识管理技术

制造业与物流服务专业

Domain-Specific Courses



Based on Legal Protocols in Global Creative Commons

超越学科界限的认知基础

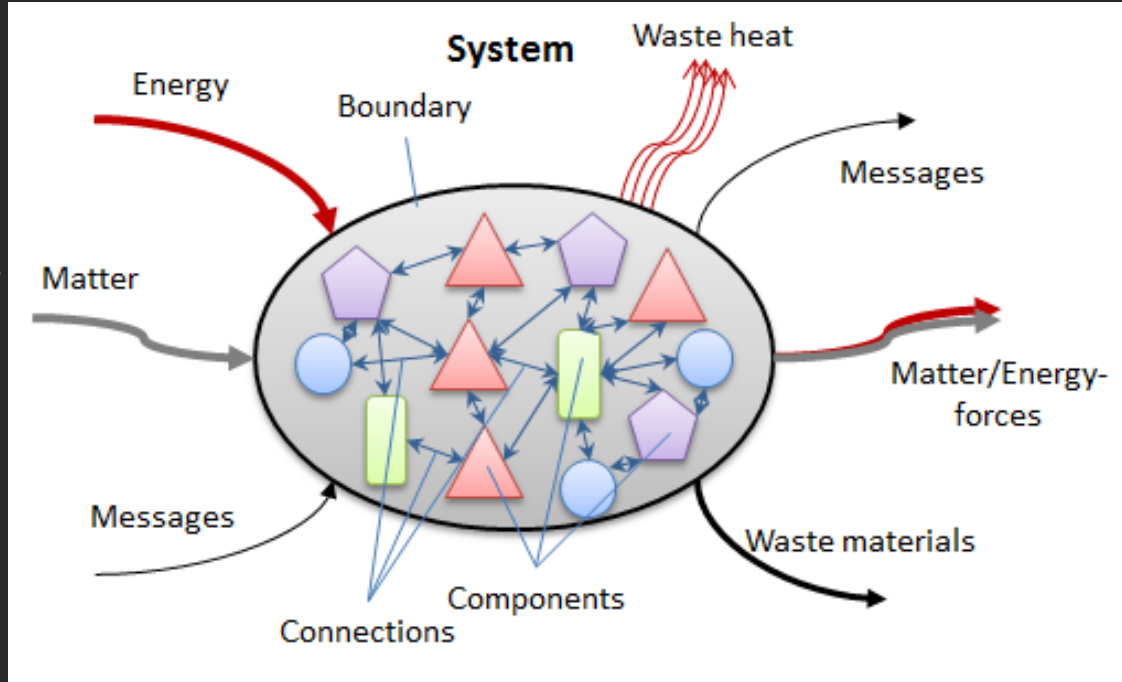
# Knowledge Container on Different Scales

Content Type	Workflow Unit	Supporting Tools
Text	Individual Activity	MediaWiki, Text Editor
Directory	Collaboration Efforts	Git, GitLab, GitHub
Micro Service	Computational Service	Docker/RTK/...
Configuration and Deployment	Development & Operations (DevOps)	Kubernetes/Trello/WeS lack
Physical Environment	Context Specific Services	Maker Space Schools

# What you should and can learn in the MEM program?



Prof. George Mobus  
University of Washington Tacoma,  
Institute of Technology



# Layered Systems

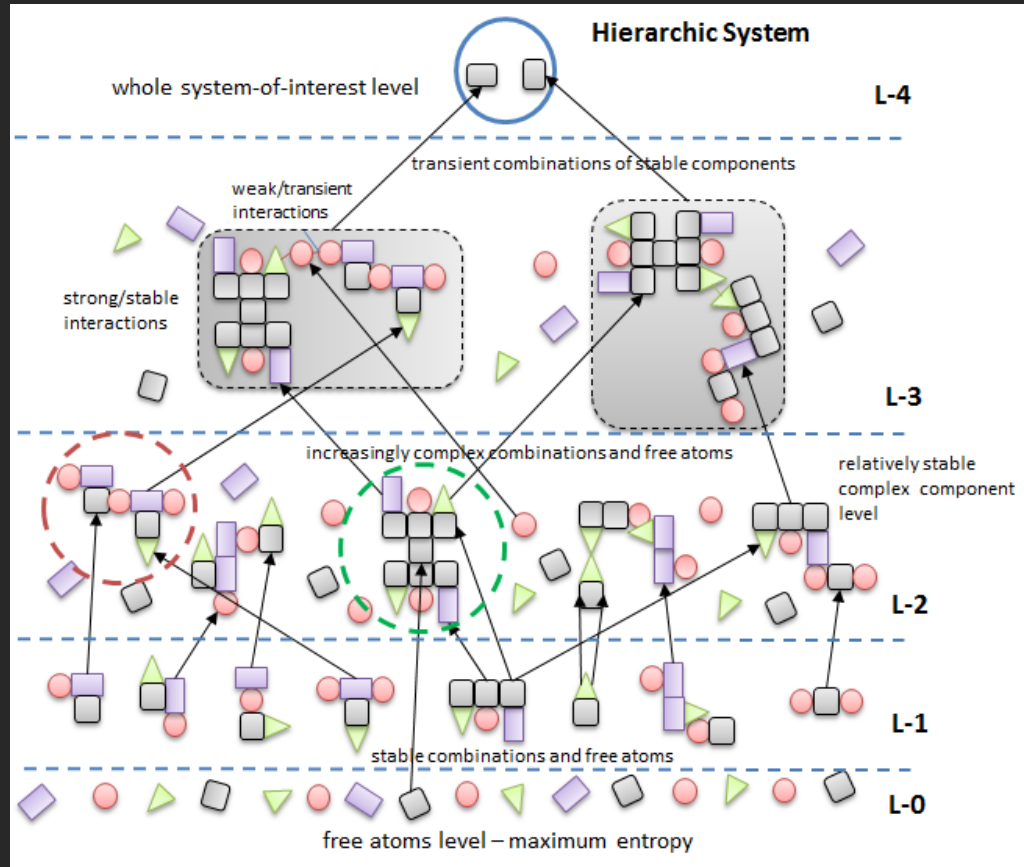
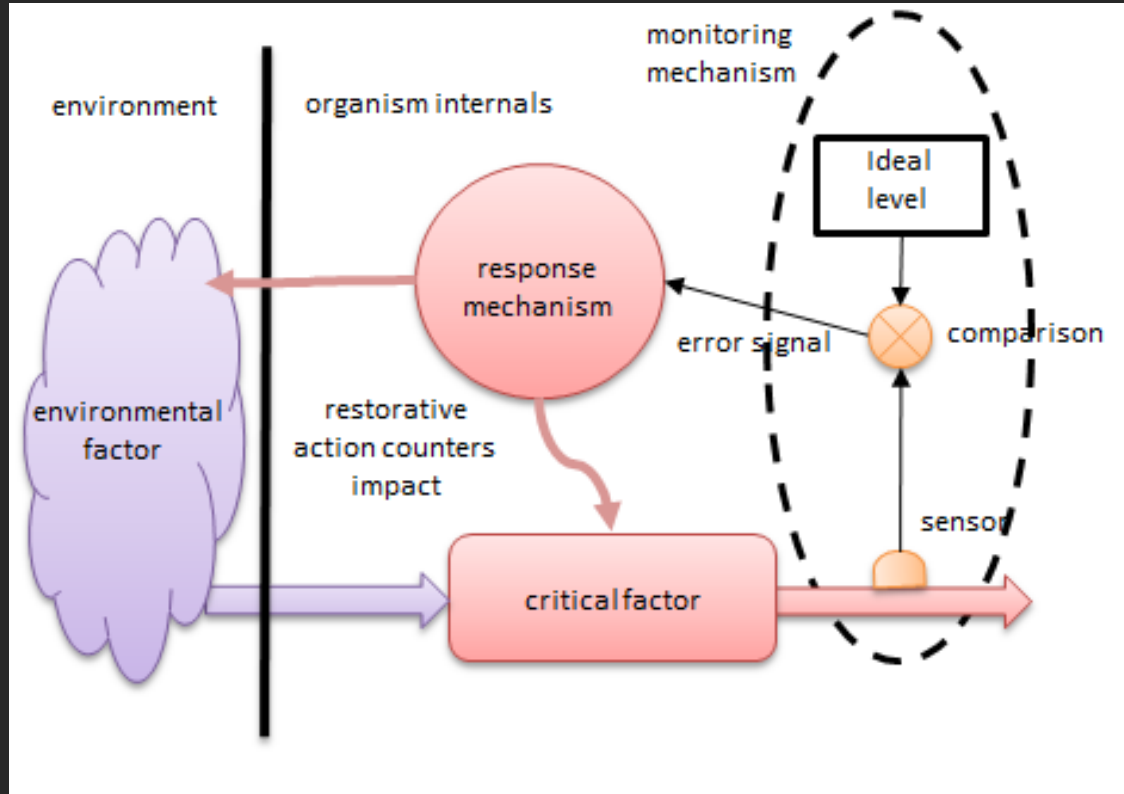


Figure 2 on the following page

[http://questioneverything.typepad.com/question\\_everything/2013/05/](http://questioneverything.typepad.com/question_everything/2013/05/)

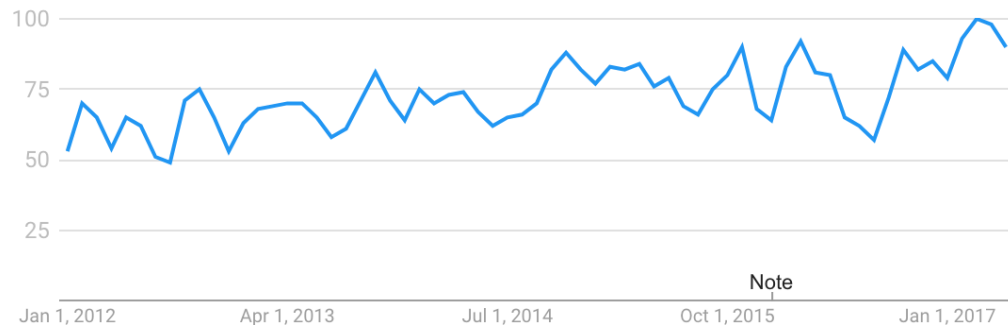
# Input-Output Systems (Measurement and Feedback)



Prof. George Mobus' Blog (Figure 3 on the following page)  
[http://questioneverything.typepad.com/question\\_everything/2013/05/](http://questioneverything.typepad.com/question_everything/2013/05/)

# Trend

Based on  “category theory” interest over past 5 years (worldwide):



## Legend:

Numbers represent search interest relative to the highest point on the chart for the given region and time.

- A value of 100 is the peak popularity for the term.
- A value of 50 means that the term is half as popular.
- Likewise a score of 0 means the term was less than 1% as popular as the peak.

## Related topics

1	Category theory - Field of study	100	<div></div>	<b>Legend</b> The most popular topics. Scoring is on a relative scale where a value of 100 is the most commonly searched topic, a value of 50 is a topic searched half as often, a value of 0 is a topic searched for less than 1% as often as the most popular topic.
2	Theory - Topic	40	<div></div>	
3	Category - Mathematics	25	<div></div>	
4	mathematics - Field of study	10	<div></div>	
5	Science - Discipline	5	<div></div>	

## Related queries

1	theory test	100	<div></div>	<b>Legend</b> The most popular topics. Scoring is on a relative scale where a value of 100 is the most commonly searched topic, a value of 50 is a topic searched half as often, a value of 0 is a topic searched for less than 1% as often as the most popular topic.
2	category theory pdf	65	<div></div>	
3	basic category theory	35	<div></div>	
4	driving theory test	30	<div></div>	
5	category theory for scientists	25	<div></div>	

---

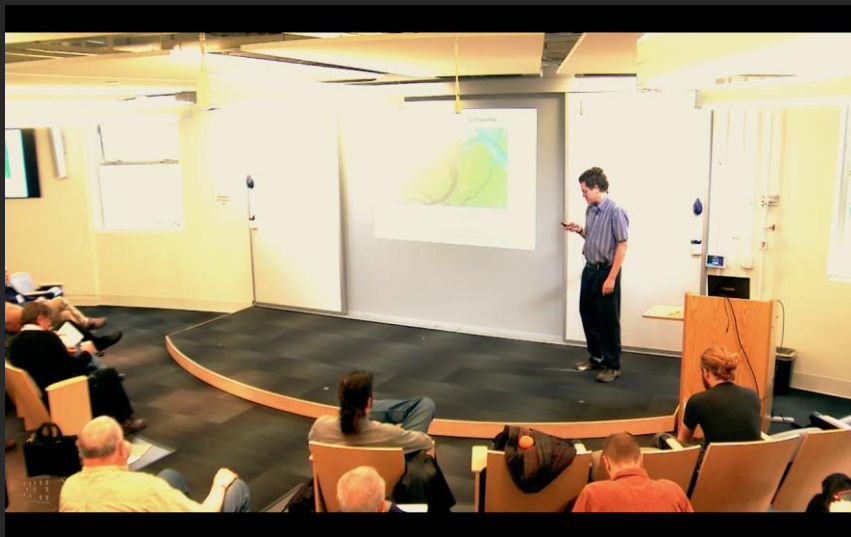
# 01. What is Category Theory?

---



# Preface

John Baez\* : ***Category Theory is the Mathematics of networks.***



[https://www.youtube.com/watch?v=lyJP\\_7ucwWo](https://www.youtube.com/watch?v=lyJP_7ucwWo)

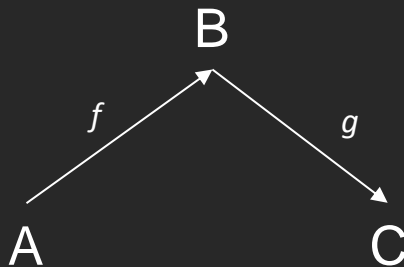
Based on Marquis, Jean-Pierre on Stanford Encyclopedia of Philosophy\*\*: *“Category theory has become an important role in contemporary mathematics and theoretical computer science. It is a powerful language, or conceptual framework, allowing us to see the universal components of a family of structures of a given kind, and how structures of different kinds are interrelated”*

Source: \*John Baez’s Stuff, <http://math.ucr.edu/home/baez/>

\*\*Jean-Pierre Marquis, <https://plato.stanford.edu/archives/win2015/entries/categorytheory/>

# Category Theory: Computation without numbers

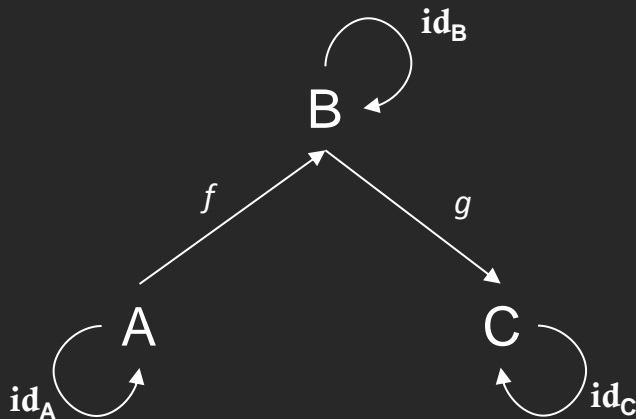
A Category is a diagram composed of Objects and Arrows



*EXAMPLE: A CATEGORY WITH 3 OBJECTS (A,B,C)  
AND TWO ARROWS (F, G)*

# What is a Category?

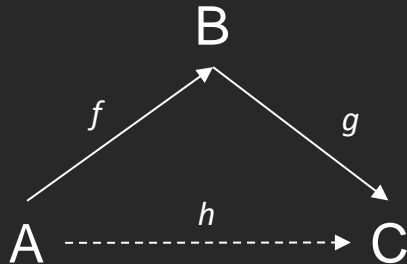
A Category must satisfy the following conditions.



1. *A COLLECTION OF OBJECTS AND A COLLECTION OF ARROWS*
2. *EVERY OBJECT  $X$  MUST IMPLICITLY HAS AN IDENTITY FUNCTION  $ID_X$*
3. *TWO OR MORE ARROWS CAN BE COMPOSED INTO ANOTHER ARROW*

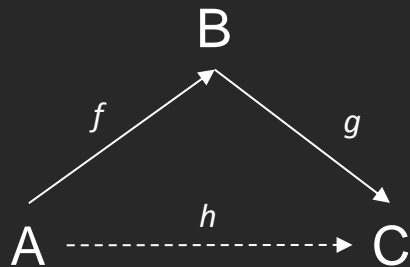
## Category is all about composing functions

*TWO ARROWS (  $f$ ,  $g$  ) CAN BE COMBINED ( **COMPOSED** ) INTO ONE ARROW  $h$*



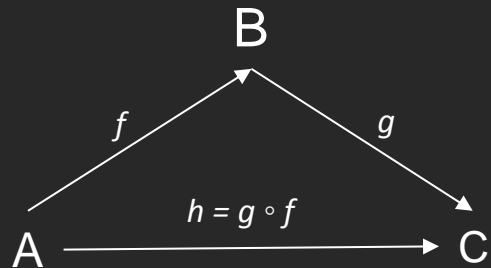
*THE ESSENCE OF CATEGORY THEORY IS TO REPRESENT THE **COMPOSABILITY** OF FUNCTIONS*

# Category Theory computes with Diagram (Arrow) Chasing



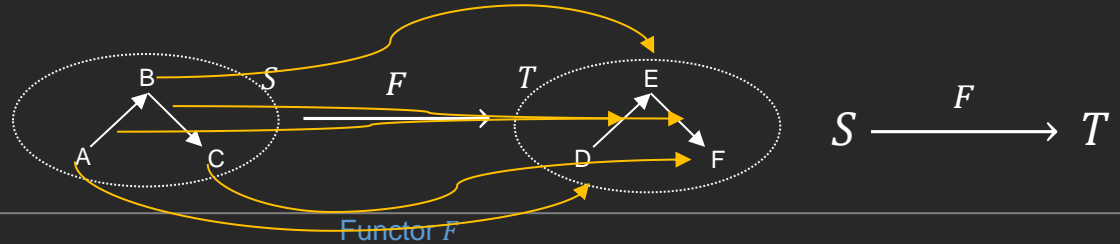
Always implies:

$f$  and  $g$  can be  
**composed** into  
 $h$   
where  
 $h = g \circ f$

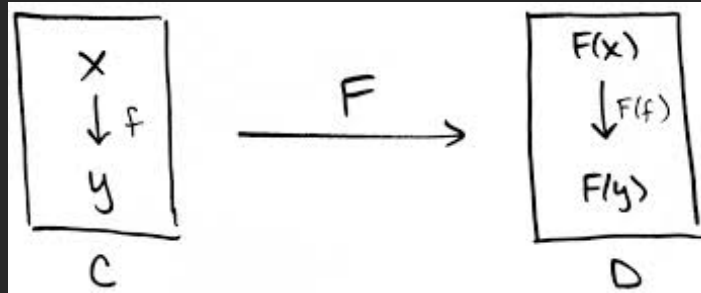


# Arrows are “mappings” between objects, categories and functors

## 2. Functor

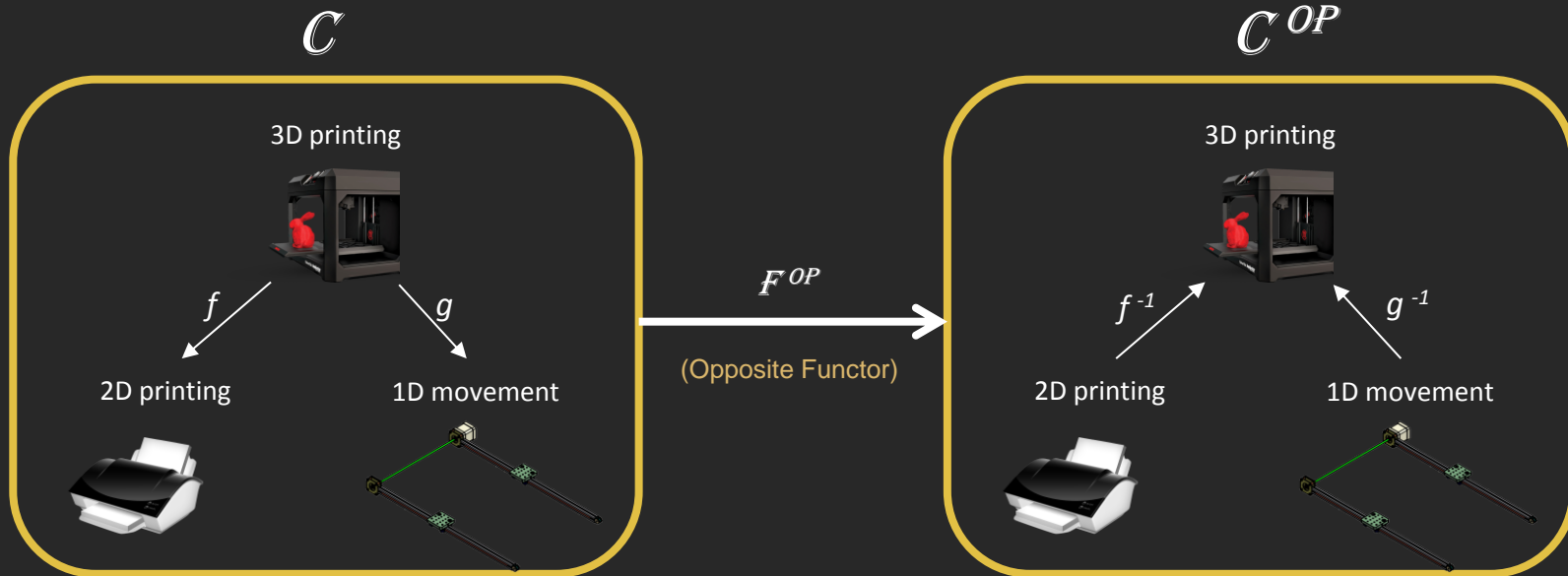


$\text{Map}[F, \text{List}[1,2,3]] =$   
 $\text{List}[F[1], F[2], F[3]]$



# How are arrows chased and composed in real life?

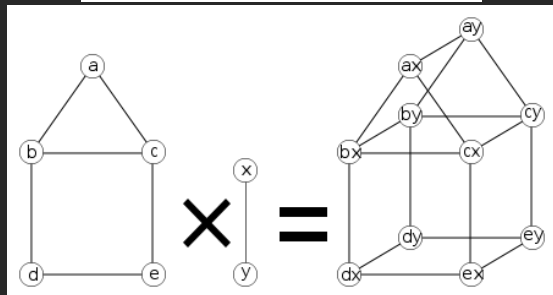
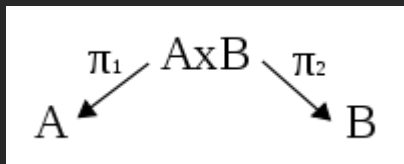
Here's another example :



# How are arrows chased and composed in real life?

Here's another example :

*PRODUCT*



$F^{OP}$

(Opposite Functor)

*SUM(CO-PRODUCT)*



$$A \oplus B = \{(1, a) \mid a \in A\} \cup \{(2, b) \mid b \in B\}.$$



---

## 03. 逻辑模型

---

## 背景

逻辑模型是本课程的重要工具之一  
本课程的很多内容都以逻辑模型的形式体现  
使用逻辑模型可以被描述为一种计算思维的应用

## 目标

让学生了解逻辑模型是甚么  
让学生了解逻辑模型的作用  
让学生了解逻辑模型与计算思维之间的关联；让学生习惯计算思维的应用

### 效果

根据学生提交的报告

评估学生对范畴论的理解  
评估学生对范畴论的接受程度

### 输出

学生需要挑选一种逻辑思维并填写其逻辑模型

自己搜索逻辑思维相关素材  
利用逻辑模型描述事件

### 过程

逻辑模型的起源  
逻辑模型是甚么  
逻辑模型的作用  
逻辑模型与计算思维之间的关联

### 输入

逻辑模型的参考文献  
经修改后的逻辑模型

## 外部因素

# 起源

20世纪70年代，逻辑模型被提出

Weiss, C.H. (1972). Evaluation Research. Methods for Assessing Program Effectiveness. Prentice-Hall, Inc., Englewood Cliffs, New Jersey

提出从给定的几个方面评估一套程序的有效性

使用者只需要将程序与逻辑模型中各个部分对应，就可以根据模型中各部分的关联验证程序的有效性

20世纪80年代，改变理论(Theory of Change)被提出

核心内容

提出描绘一套方案它的长期，中期和短期之间的效果连结关系，从而解释方案落实所带来的变化过程

以上

[注] 方案的每一阶段都会产生新的变化，变化的效果会成为下一个阶段的前提，形成效果连结关系

对已有的变化，以效果路线规划，显示每个效果与其他效果的逻辑关系与时间流程

[注] 已有变化->效果->效果->效果->目标

透过回溯效果路线规划，可以解释一个阶段为甚么要求上个阶段达到甚么样的效果

创新点

区别期望效果与实际效果的差异

[注] 现实效果->过渡效果->目标效果

需要自己填补中间的过程

要求人先形塑其所想要的效果

[注] 现实效果->过渡效果->目标效果

要求利益相关者达到前提的效果

# 逻辑模型的内容

根据参考文献，逻辑模型最简单的形式包括4个部分

## 结果

[例] 根据学生的作业，可以验证学生的对课程的吸收能力，并判断课程是否成功

## 输出

[例] 学生的作业

## 活动

[例] 老师授教，学生完成作业

## 输入

[例] 教堂，老师，学生，教材

# 逻辑模型的内容

基于最简单的形式，我们进行了修改

逻辑模型

## 背景

需要分析或者计划的的事物在甚么一个环境上出现  
在这个环境以外的因素我们不需要考虑或不考虑

## 目标

所有事情都必然存在目标，因此我们必须定明目标  
[例] 我要减肥

## 效果

目标需要以某种方式被验证，换言之是达到甚么具体效果  
[例] 减10kg

## 输出

这个效果体现在甚么东西上面  
[例] 做了某些事情后的我

## 过程

为了产生这个输出，需要完成甚么步骤  
[例] 白天骑车上班，晚上跑步，持续一个月

## 输入

为了支持过程的进行，我们需要一些东西来支持  
[例] 自行车，运动鞋，我

## 外部因素

有甚么不可控因素会导致方案的实行受影响  
[例] 赶工作进度

# 逻辑模型的作用

- 对于**现有的项目方案**，活动计划等等都可以利用逻辑模型来重新描述
  - 清晰方案内部的逻辑关系
  - 发现方案内部的逻辑漏洞
- 对于**待解决的问题**也都可以利用逻辑模型来重新描述
  - 依循逻辑模型的字段明显问题的核心要素
  - 依循逻辑模型的思路判断是否存在可行方案
  - 依循逻辑模型的思路判断订立具体方案

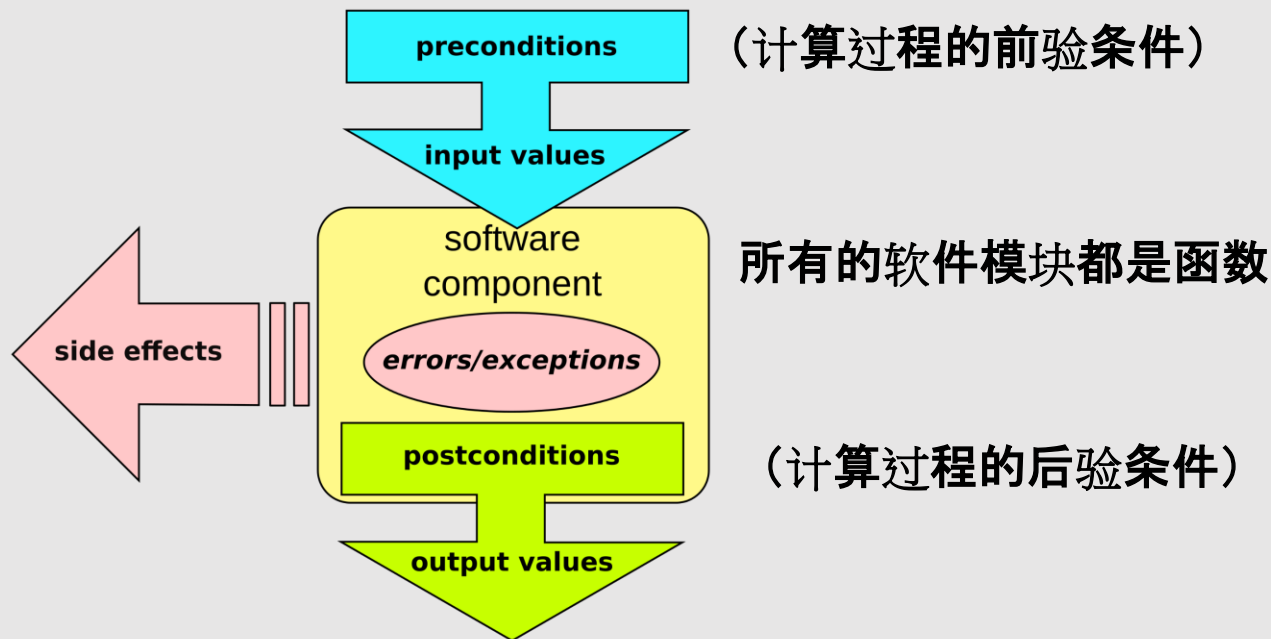
# 逻辑模型与计算思维之间的关联

在计算机软件工程中，我们需要进做软件系统设计  
把大型软件系统按功能拆解成低耦合的模块

[例] 微信：用户个人信息，用户聊天记录，用户  
朋友圈等等

每个模块都对和某个模块对话，

# 何谓函数? What is function?

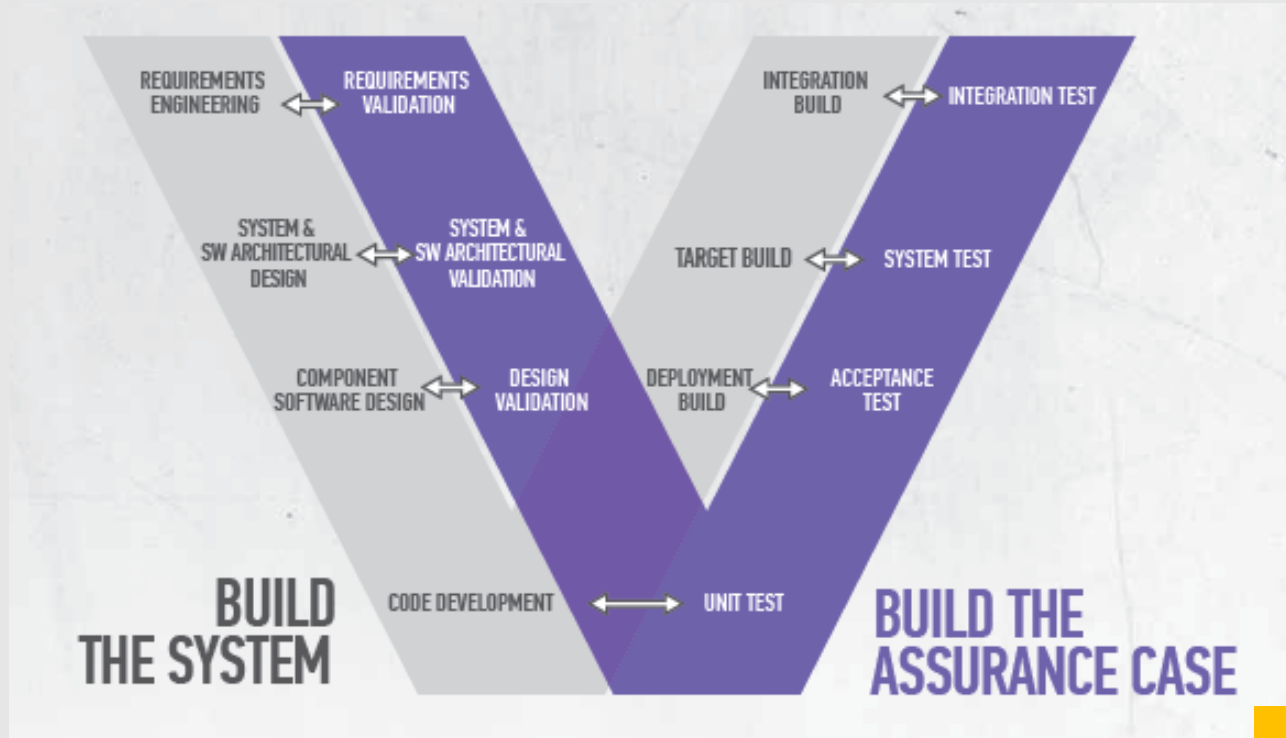


[https://en.wikipedia.org/wiki/Design\\_by\\_contract](https://en.wikipedia.org/wiki/Design_by_contract)

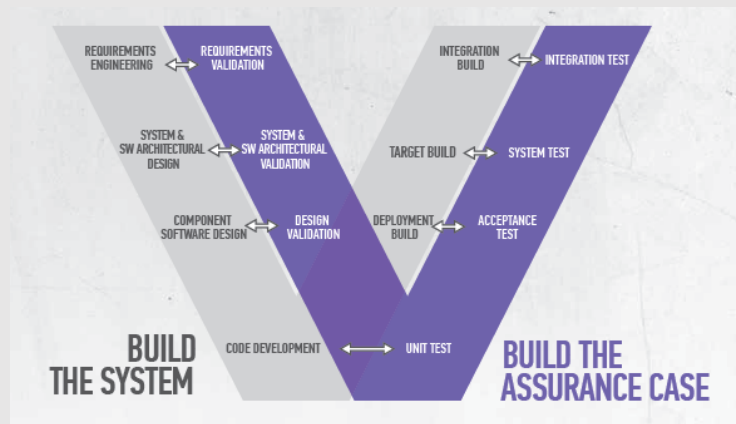


► How do you know that you are **Correct**?

## Software Verification and Validation (V&V)



# ► How do you know that you are **Correct**?

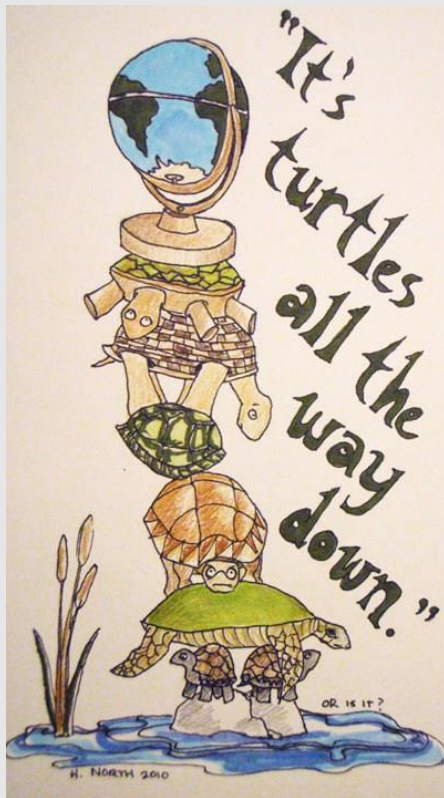
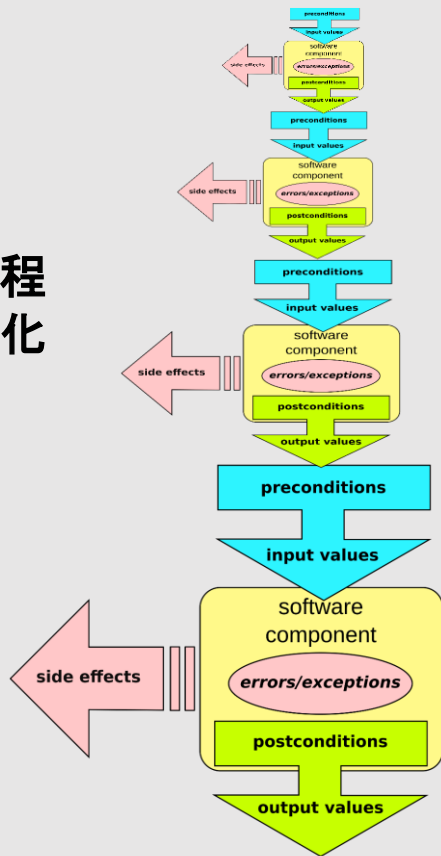


**Verification: Build the product based on “contract”  
(do the **right** thing)**

**Validation: Build to deliver “desirable” outcomes  
(do the things **right**)**


# 如何检验复杂系统？

检验过程  
的迭代化



# How do you know that you are right? [www.nand2tetris.org](http://www.nand2tetris.org)

From NAND to Tetris  
*Building a Modern Computer From First Principles*



[Home](#)  
[Projects](#)  
[Book](#)  
[Software](#)  
[Media](#)  
[Cool Stuff](#)  
[Terms](#)  
[Q&A](#)  
[About](#)

## Project 1: Logic Gates

**Background**

A typical computer architecture is based on a set of elementary logic gates like And, Or, etc., as well as their bit-wise versions And16, Or16, etc. (in a 16-bit machine). This project engages you in the construction of a typical set of elementary gates. These gates form the elementary building blocks from which more complex chips will be later constructed.

**Objective**

Build all the logic gates described in Chapter 1 (see list below), yielding a basic chip-set. The only building blocks that you can use in this project are primitive Nand gates and the composite gates that you will gradually build on top of them.

**Chips**

Chip (HDL)	Function	Test Script	Compare File
Nand	Nand gate (primitive)		
Not	Not gate	Not.tst	Not.cmp
And	And gate	And.tst	And.cmp
Or	Or gate	Or.tst	Or.cmp
Xor	Xor gate	Xor.tst	Xor.cmp
Mux	Mux gate	Mux.tst	Mux.cmp
DMux	DMux gate	DMux.tst	DMux.cmp
Not16	16-bit Not	Not16.tst	Not16.cmp

Project 1 web site

And.hdl ,  
And.tst ,  
And.cmp files

# How do WE know that you Know?



XLP的重要原則：設計合約與驗證。

## 挑戰設計方

- 設計課程合約。
- 建立迭代的驗證體系，通過驗證來知道任務執行方是否知道。

## 任務執行方

- 按照課程合約，接受正確的輸入，輸出出版物。
- 出版物是驗證執行方是否知道的重要手段。



# 参考文献

[1] Alter, C., & Murty, S. (1997). Logic modeling: a tool for teaching practice evaluation. Journal of Social Work Education, 33(1), 103-117. [2] Logic Model – Wikipedia  
[https://en.wikipedia.org/wiki/Logic\\_model](https://en.wikipedia.org/wiki/Logic_model)

---

## 04. Wiki和Git

---

## 背景

wiki和git是当下极具实用价值的技术  
wiki和git是本课程主要的协作管理技术  
wiki和git是本课程的技术支持，同时他们也是计算思维的案例

## 目标

让学生能够掌握实用技术  
让学生在课程中能够高效协作  
借用wiki和git技术以及相关工具的使用深度学习了解计算思维

### 效果

根据逻辑模型和每日心得的内容迭代，评估学生的理解和收获状态  
根据github上作业的质量，评估学生能力

### 输出

wiki上的每日心得，逻辑模型  
github上上传作业

### 过程

wiki,git的发展历史  
wiki和git是什么  
wiki与wikipedia以及mediawiki的区别  
toyhouse,github以及git软件gitkraken用法  
计算思维与wiki,git的联系

### 输入

git技术的客户端软件gitkraken \* wiki, git的参考文献  
toyhouse,github网站，以及使用的参考文献  
以往课程的toyhouse和GitHub使用案例

## 外部因素



# wiki

- **wiki**

- wiki 是一种在网络上开放并且可以供多人协同创作的超文本系统。

- **mediawiki**

- MediaWiki是一套基于网络的Wiki引擎，维基媒体基金会的所有项目乃至众多wiki网站皆采用了这一软件。MediaWiki软件最初是为维基百科所开发的。

- **wikipedia**

- 维基基金会在mediawiki之上建立的应用



- **wiki与wikipedia以及mediawiki的区别**

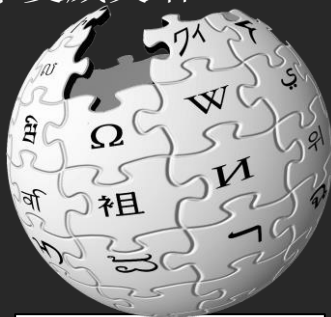
- wiki是技术，mediawiki是工具，wikipedia是在mediawiki工具基础上的应用，维基基金会有众多基于mediawiki工具和wiki技术的应用
- 我们使用的toyhouse.cc网站是基于mediawiki工具和wiki技术搭建的自用wiki平台，而wikipedia是属于维基基金会的wiki平台。

- Wiki: <https://zh.wikipedia.org/wiki/Wiki>      wikipedia: <https://zh.wikipedia.org/wiki/维基百科>

# wiki

## • wiki的发展史

- Wiki软件是由软件设计模式社群发展出来，用来书写与讨论模式语言。沃德·坎宁安于1995年3月25日成立了第一个Wiki网站：WikiWikiWeb，用来补充他自己经营的软件设计模式网站。他发明了Wiki这个名字以及相关概念，并且制作了第一个Wiki引擎。坎宁安说自己是根据檀香山的Weekee Weekee(快点快点)公车取名的。这是他到檀香山学会的第一个夏威夷语。



**WIKIPEDIA**  
*The Free Encyclopedia*

# 计算思维与wiki的联系

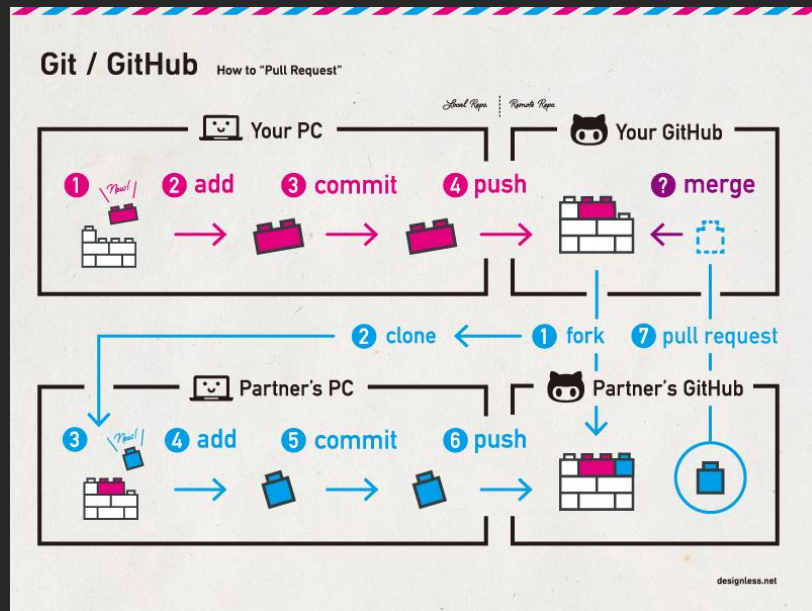
wiki具有多人协同创作，有历史记录，可回溯，修改内容比对的功能，从而可以提供时间，空间，可信度的证据。

wiki里强调参考文献的重要性，同时也是计算思维中的判断依据。

# git

## • 什么是git

- Git是一款免费、开源的分布式版本控制系统，用于敏捷高效地处理任何或小或大的项目。
- Git是一个开源的分布式版本控制系统，可以有效、高速的处理从很小到非常大的项目版本管理。Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件。





## • Git 的发展史

- Git 诞生于一个极富创新的年代。
- Linux 内核开源项目有着为数众多的参与者。绝大多数的 Linux 内核维护工作都花在了提交补丁和保存归档的繁琐事务上（1991—2002年间）。到 2002 年，整个项目组开始启用一个专有的分布式版本控制系统 BitKeeper 来管理和维护代码。
- 到了 2005 年，开发 BitKeeper 的商业公司收回了 Linux 内核社区免费使用 BitKeeper 的权力。这就迫使 Linux 开源社区（特别是 Linux 的缔造者 Linus Torvalds）基于使用 BitKeeper 时的经验教训，开发出自己的版本系统。他们对新的系统制订了若干目标：
  - 速度
  - 简单的设计
  - 对非线性开发模式的强力支持（允许成千上万个并行开发的分支）
  - 完全分布式
- 有能力高效管理类似 Linux 内核一样的超大规模项目（速度和数据量）
- 2005 年以来，Git 成熟完善，它的速度飞快，极其适合管理大项目，有着令人难以置信的非线性分支管理系统
  - [git简史](https://git-scm.com/book/zh/v2/起步-Git-简史)：https://git-scm.com/book/zh/v2/起步-Git-简史

# git

- **git与计算思维的联系**

- 基于git 分布式管理系统的网站GitHub在像wiki支持多人共编，有历史版本的基础上，有分支结构，支持本地仓库。从而会有参与者的时间空间，和内容修改记录。从而见证可信度。

---

## 05. Markdown

---

## 背景

markdown是当下流行的轻量级标记语言  
markdown是本课程重要的内容写作语言  
markdown是计算思维的典型范例

## 目标

让学生掌握流行的写作方法  
让学生高效的进行内容协作和写作  
让学生在使用markdown的过程中学习和理解计算思维

### 效果

根据学生对  
markdown的使用熟  
练程度和排版质量  
评估学生的理解程  
度

### 输出

在GitHub上用  
markdown格式写作业

### 过程

markdown是什么  
markdown的发展  
markdown与计算思  
维的关系

### 输入

让学生在使用  
markdown的过程中  
学习和理解计算思  
维  
markdown语法

## 外部因素



# markdown

- 什么是markdown?

- Markdown 是一种轻量级标记语言，它允许人们“使用易读易写的纯文本格式编写文档，然后转换成有效的XHTML(或者HTML)文档”。这种语言吸收了很多在电子邮件中已有的纯文本标记的特性。

- markdown与计算思维的关系

- markdown作为一种标记语言，和很多标记语言在交集的部分同构，像是HTML，wiki的语法。都是在用特定的语法标签进行格式化排版。

Q&A