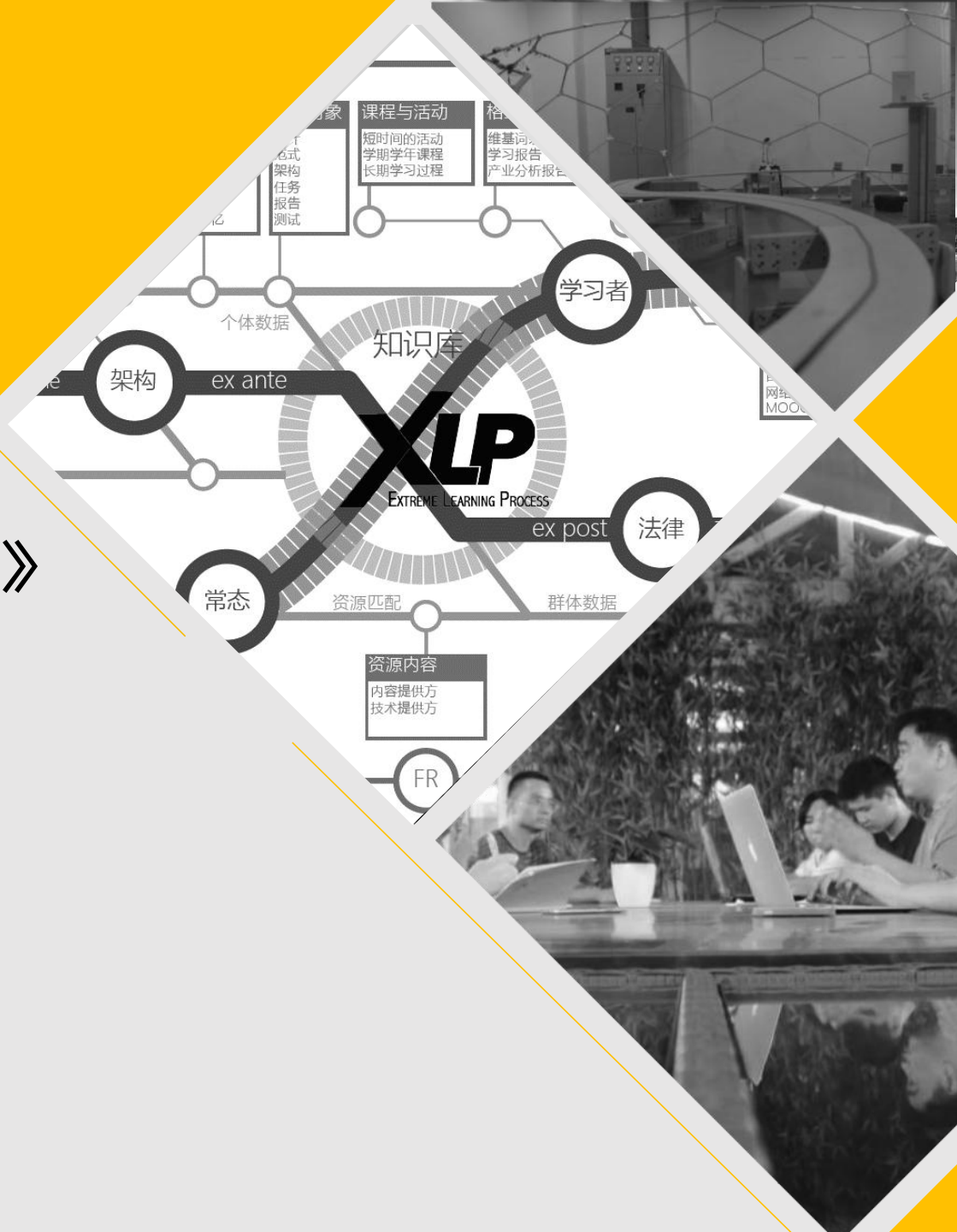


《超越学科界限的认知基础》

第三讲

自然科学的认知进程

主讲老师：顧學雍

助教：陶仁鹏

课程wiki主页：toyhouse.cc

2017年秋 周二 13：30~16：50

地点：李兆基大楼B643

Colophone: 扉页

Usage and Copyright Notice:

We provide 16 such presentations and many additional lectures by invited speakers

Each presentation is designed to support about 3 hours of classroom or self-study instruction.

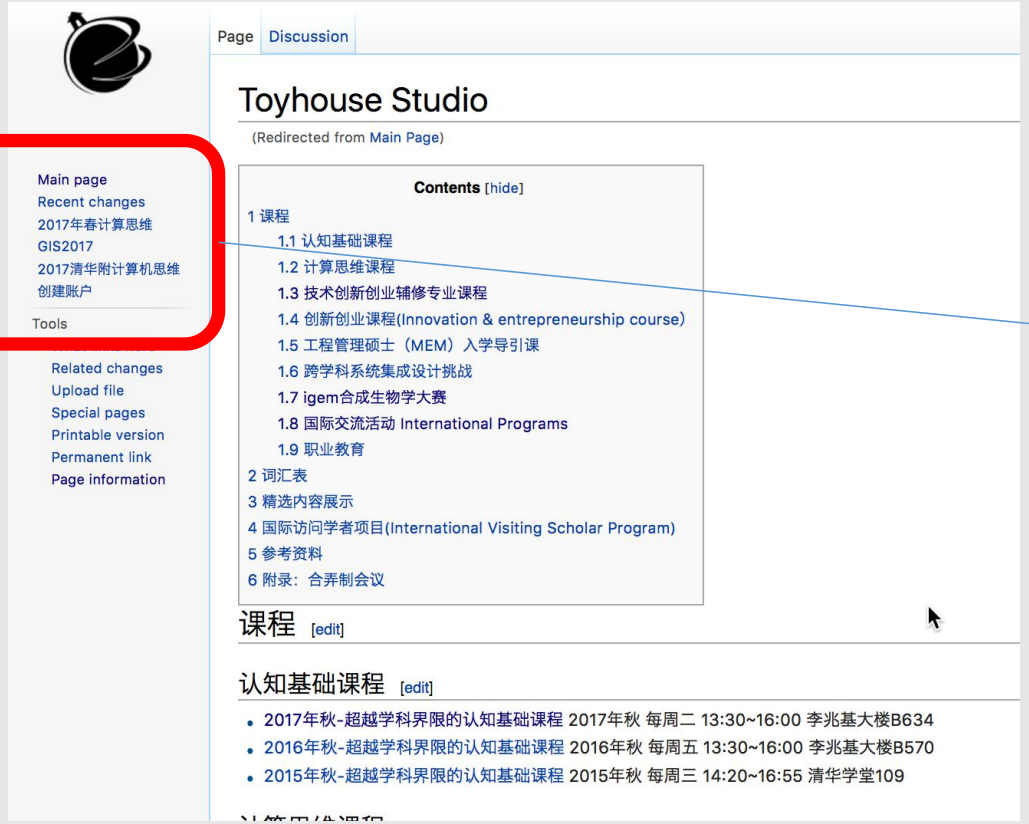
You are welcome to use or edit this presentation as you see fit for instructional and non-commercial purposes.

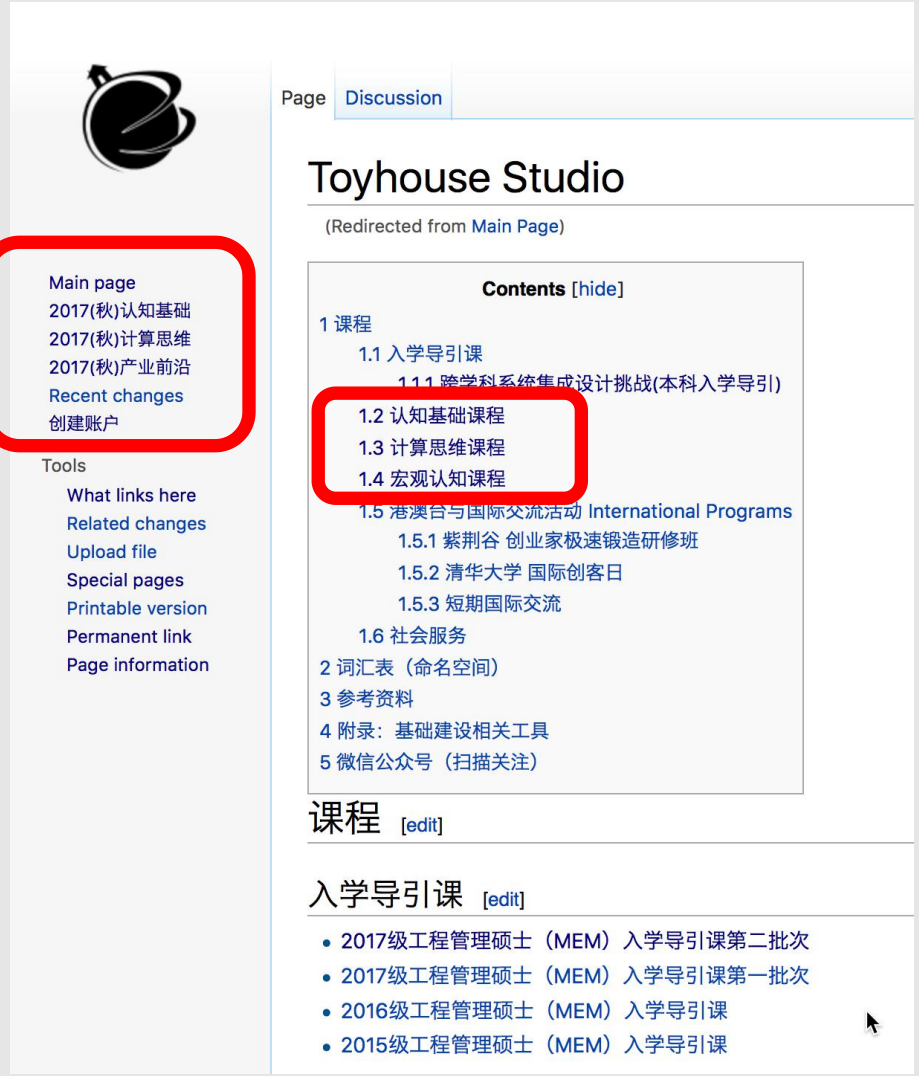
If you use our book and course materials, please include a reference to www.nand2tetris.org and toyhouse.cc

If you have any questions or comments, please write us at koo0905@gmail.com

超限學習過程

Toyhouse 改版之后的结构





认知技能来自隐喻

- 本体隐喻：脑中的概念与实体世界的联系
  - 已熟悉的词汇 （如桌椅、门窗、飞禽走兽。。。）
- 空间隐喻：各物品或概念单元的时空定位关系
  - 物件中的上下左右、事件发生的前后（偏序关系）
  - 社会地位、亲属联系的高低大小关系
- 系统隐喻：整套物品（事件）的结构类似性
  - 机器人对应与木流牛马
  - 计算机的作业系统对应与国家的宪章

Carlo Rovelli 物理学的七堂课

Seven Brief Lessons on Physics 第五章翻译文稿

粒空间 （Grains of Space） [edit]

尽管依旧艰难依旧困惑依旧有解决不了的问题，但我已经比我们之前所能描绘的物理现象有了更清晰的轮廓。所以，按理说我们应该满意，但是我们还不能。

在我们对物理学世界理解的核心中有一个悖论。二十世纪已经遗留给了我们两大物理学珍宝：相对论和量子力学。相对论推动了宇宙论的发展，还有天体物理学，以及引力波，黑洞等很多其他方面的研究的发展。量子力学为原子物理，核物理，基本粒子物理，凝聚态物理以及许多许多方面提供了理论根基。两个理论的应用，是我们今天科技的基础，并已经改变了我们的生活方式。但这两个理论至少在现在的形式下一定不都是对的，因为它们相互矛盾。

一个早晨听了相对论的演讲，下午听了量子力学的演讲的大学生如果得出，他的老师是傻子，或者这两个教授已经至少一个世纪都没交流过的结论，是可以被原谅的。在早晨，这个世界是被卷曲的但依旧处处连续的。在下午，这世界就是一个平的空间，处处有量子能级。

一个大学生早晨听了相对论的演讲，下午听了量子力学的演讲，如果他得出，他的老师是傻子，或者这两个教授已经至少一个世纪都没交流过的结论，是可以被原谅的。在早晨，这个世界是被卷曲的但依旧处处连续的。在下午，这世界就是一个平的空间，处处有量子能级。

悖论在于这两个理论在理论实践中都发挥的相当完美。自然正使我们表现的像一个老年法师去调解两个争论不休的男人。已经听完了第一个人的论述，法师说：“你是对的。”，第二个人坚持要陈述观点，法师听完他表述之后说到：“你也是对的。”在隔壁屋里，偶然间听到这谈话的法师的妻子说道：“但他们不可能都是对的啊。”法师沉思了，并且在得出“你们都是对的”之前来回纠结着。

机器翻译与人翻译的对比

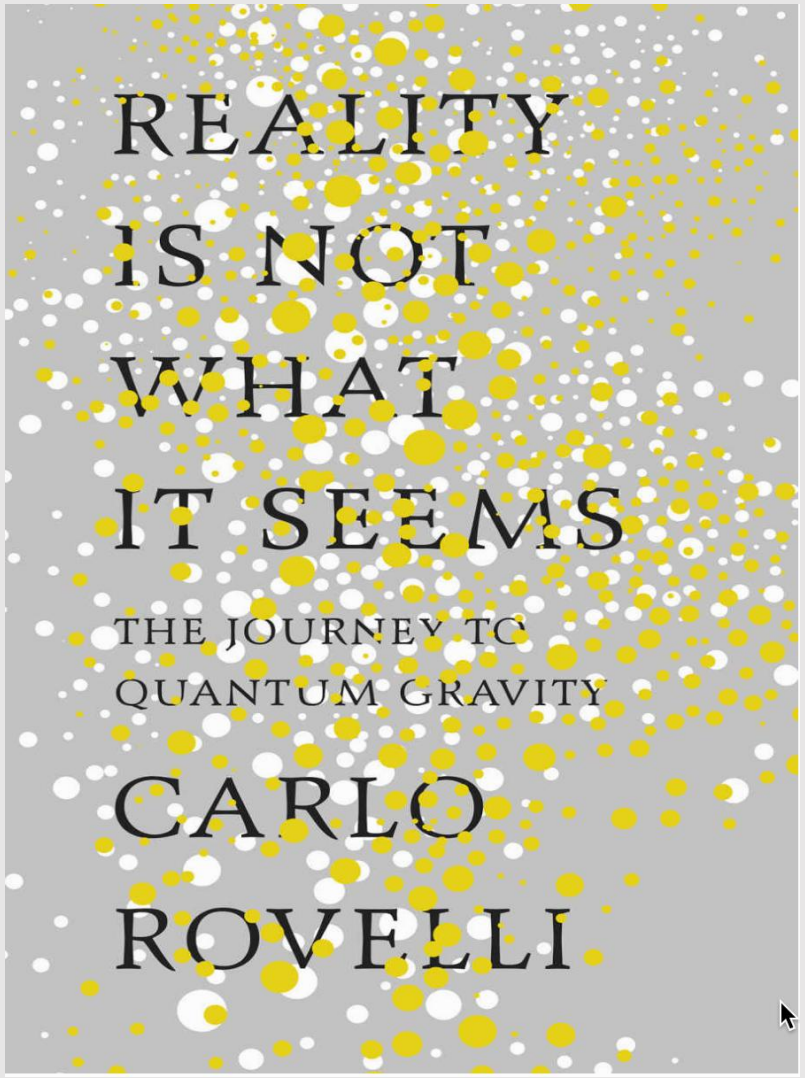
Seven Brief Lessons on Physics 第五章翻译文稿

粒空间 （Grains of Space） [edit]

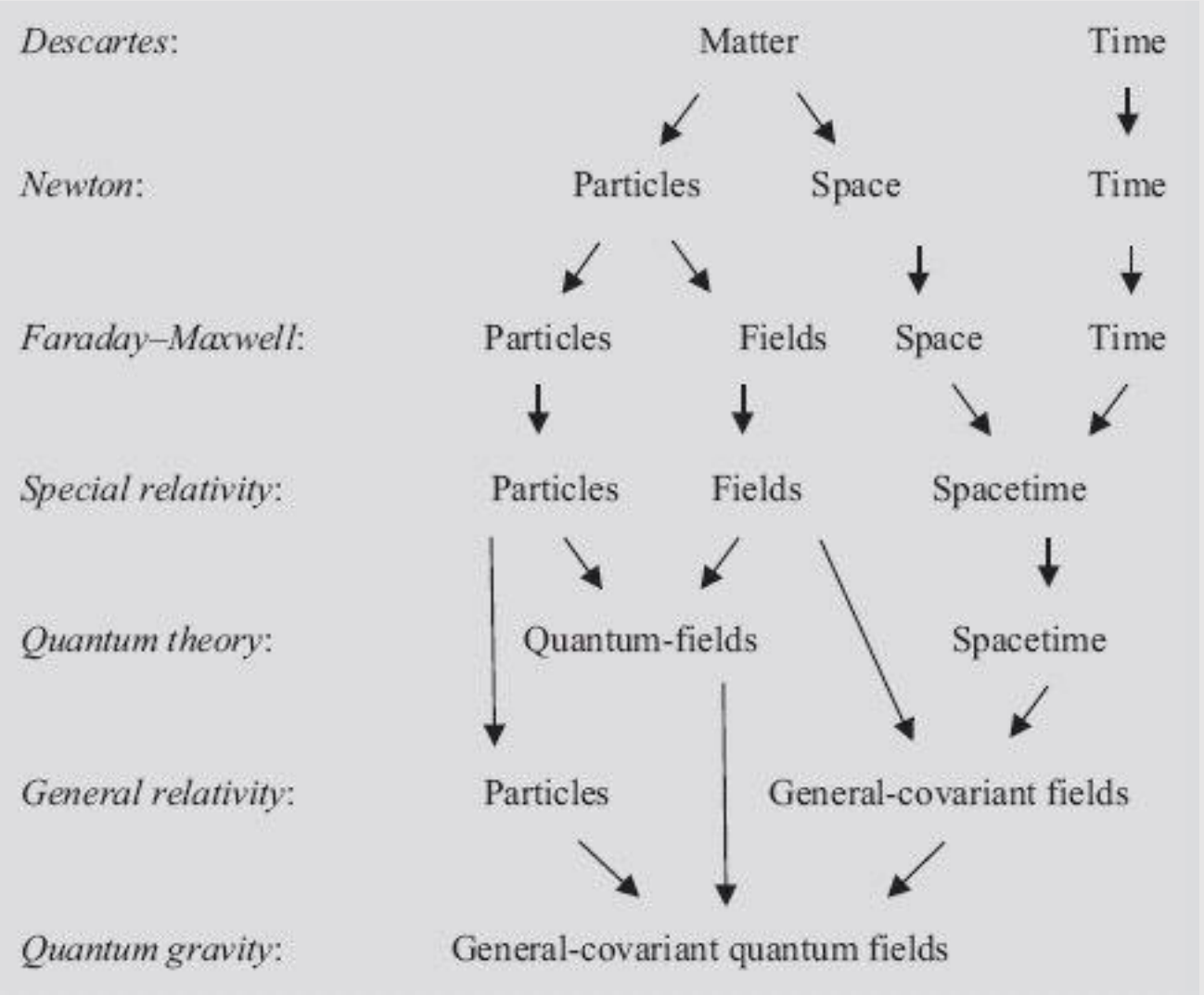
尽管依旧艰难依旧困惑依旧有解决不了的问题，但我已经比我们之前所能描绘的物理现象有了更清晰的轮廓。所以，按理说我们应该满意，但是我们还不能。

“Despite certain obscurities, infelicities and still unanswered questions, the physics I have outlined provides a better description of the world than we have ever had in the past. So we should be quite satisfied. But we are not.”

物理学家 Carlo Rovelli 对时空结构的解说

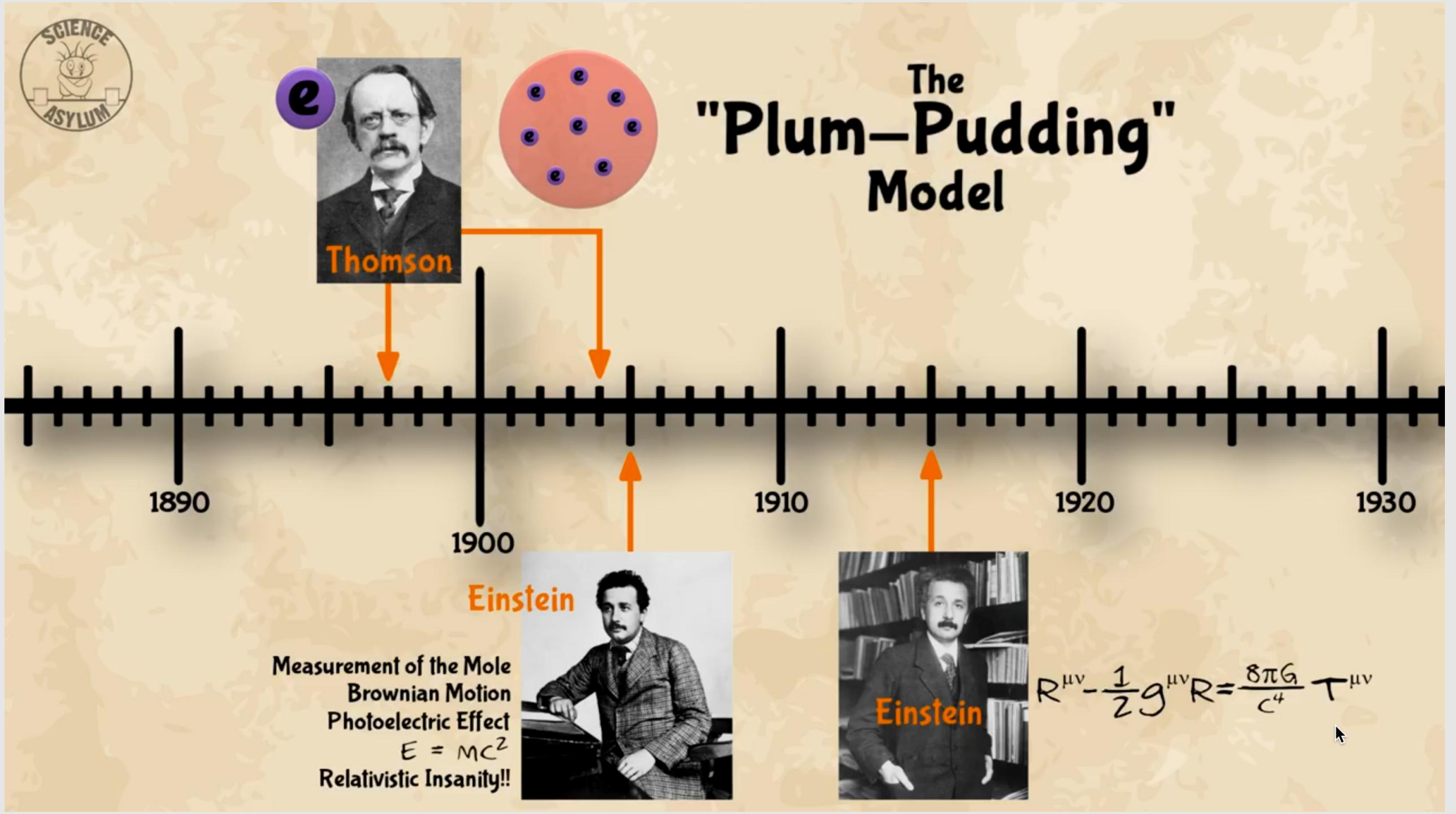


Carlo Rovelli的 Class Diagram (知识图谱 / 信息结构分类图)



Covariant Loop Quantum Gravity: An Elementary Introduction of Quantum Gravity Theory and Spinfoam Theory by Rovelli and Vidotto

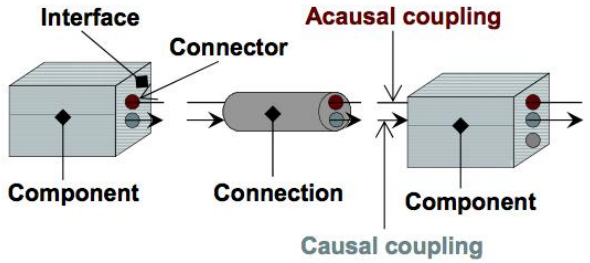
时间顺序提供的认知线索：人物史、机构史、科技史



影像来源：Youtube 上的 “what does an atom really look like?”

可计算图：Computable Diagrams

Software Component Model



A component class should be defined *independently of the environment,* very essential for *reusability*

A component may internally consist of other components, i.e. *hierarchical* modeling

Complex systems usually consist of large numbers of *connected* components

Physical Connector

- Classes Based on Energy Flow

Domain Type	Potential	Flow	Carrier	Modelica Library
Electrical	Voltage	Current	Charge	`Electrical.Analog`
Translational	Position	Force	Linear momentum	`Mechanical.Translational`
Rotational	Angle	Torque	Angular momentum	`Mechanical.Rotational`
Magnetic	Magnetic potential	Magnetic flux rate	Magnetic flux	
Hydraulic	Pressure	Volume flow	Volume	`HyLibLight`
Heat	Temperature	Heat flow	Heat	`HeatFlow1D`
Chemical	Chemical potential	Particle flow	Particles	Under construction
Pneumatic	Pressure	Mass flow	Air	`PneuLibLight`

Mathematical Models: A Sketch for the Philosophy of Math (1981)

Saunders Mac Lane从数学文献中所总结出的各种动词片语

Counting	Arithmetic, Number Theory
Measuring （度量）	Real Numbers, Calculus, Analysis
Shaping （分辨形体）	Geometry, Topology
Forming （成形）	Symmetry, Group Theory
Estimating （估测）	Probability, Measure Theory
Moving （移动）	Mechanics, Calculus, Dynamics
Calculating （计算）	Algebra, Numerical Analysis
Proving （证明）	Logic
Puzzling （解谜、加密）	Combinatorics, Number Theory
Grouping （聚类）	Set Theory, Combinatorics

http://links.jstor.org/sici?sici=0002-9890%28198108%2F09%2988%3A7%3C462%3AMMASFT%3E2.0.CO%3B2-D

词汇的比较 （Saunders Mac Lane vs. Modelica）

Counting	Arithmetic, Number Theory
Measuring （度量）	Real Numbers, Calculus, Analysis
Shaping （分辨形体）	Geometry, Topology
Forming （成形）	Symmetry, Group Theory
Estimating （估测）	Probability, Measure Theory
Moving （移动）	Mechanics, Calculus, Dynamics
Calculating （计算）	Algebra, Numerical Analysis
Proving （证明）	Logic
Puzzling （解谜、加密）	Combinatorics, Number Theory
Grouping （聚类）	Set Theory, Combinatorics

Physical Connector

- Classes Based on Energy Flow

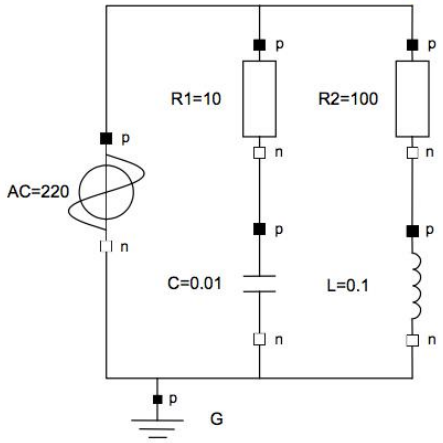
Domain Type	Potential	Flow	Carrier	Modelica Library
Electrical	Voltage	Current	Charge	`Electrical.Analog`
Translational	Position	Force	Linear momentum	`Mechanical.Translational`
Rotational	Angle	Torque	Angular momentum	`Mechanical.Rotational`
Magnetic	Magnetic potential	Magnetic flux rate	Magnetic flux	
Hydraulic	Pressure	Volume flow	Volume	`HyLibLight`
Heat	Temperature	Heat flow	Heat	`HeatFlow1D`
Chemical	Chemical potential	Particle flow	Particles	Under construction
Pneumatic	Pressure	Mass flow	Air	`PneuLibLight`

可计算图形语意：Computable Semantics

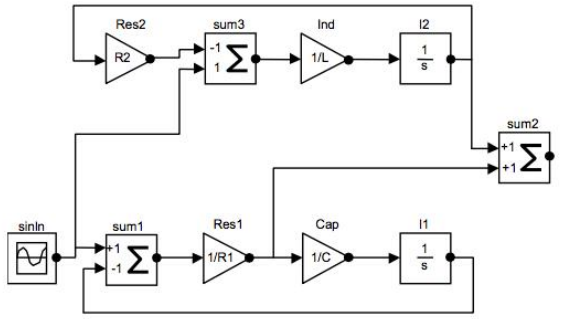
Modelica vs Simulink Block Oriented Modeling Simple Electrical Model

Modelica:
Physical model – easy to understand

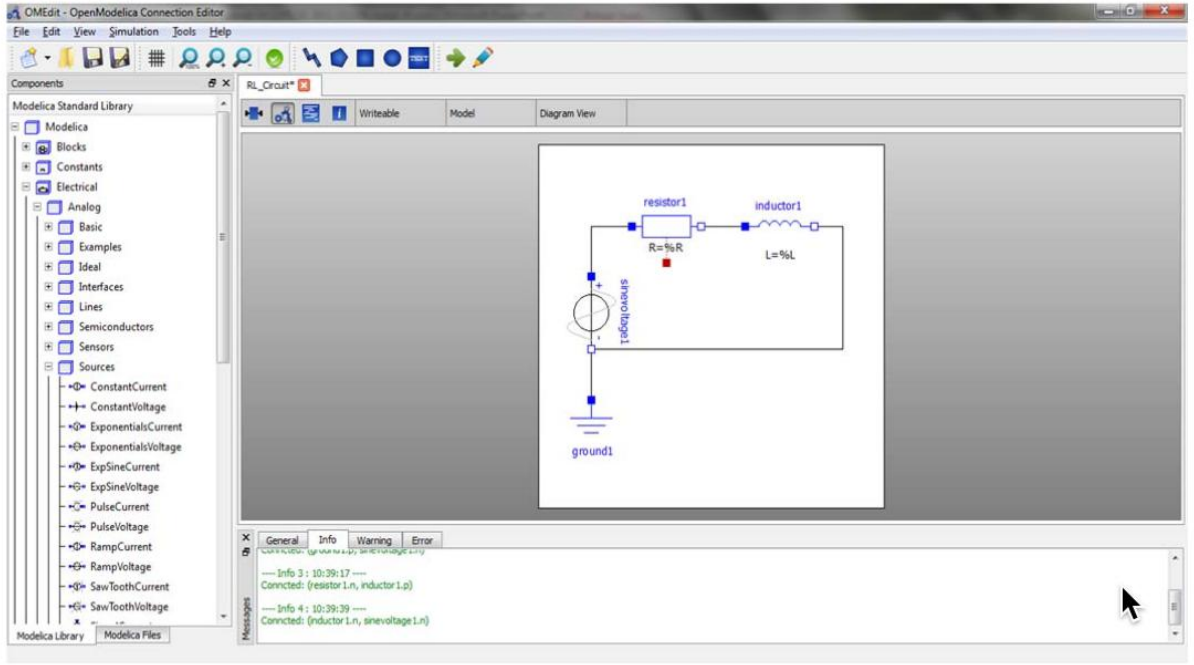
Keeps the physical structure



Simulink:
Signal-flow model – hard to understand



Graphical Modeling - Using Drag and Drop Composition



面向对象的设计概念 （范畴论的应用）

Object-Oriented Design Concepts

CSE301

University of Sunderland

Harry R. Erwin, PhD

提供一套形式化的知识编码工具

Why Object-Oriented Design?

- Object-oriented designs are easier to define initially.
  - requirements are easier to understand since you usually have a real-world model to work from,
  - more realism in the expectations of others,
  - debugging is easier.
- Object-orientation is better for many real-world applications.

在物联网世界的应用

Real-World Applications

- IT makes good sense in two contexts:
  - Where it allows you to do things you couldn't do before, and
  - When automation pays off in increased efficiency.
- In both cases, cost-effective applications allow you to:
  - Allocate resources efficiently
  - Monitor performance
  - Respond to unusual time-critical demands
  - Handle failures

描述知识的度量衡与“容错”机制

How Does This Happen?

- Allocate resources efficiently
  - By monitoring their usage and reserve status
- Monitor performance
  - By tracking what is going on in the real world
- Respond to unusual time-critical demands
  - By providing the tools needed to adapt
- Handle failures
  - By designing in assessment and recovery mechanisms

All these features are dependent on having a model of how the system actually operates.

知识模型的建构策略 （Composability）

Models

- You can have two kinds of models:
  - Something you use to design the system, or
  - Something in the system that acts as a model.
- You need the latter if system requirements are likely to change or are unpredictable in detail
- Internal system models predict how the system will respond to demands. For robustness, you design in data collection to correct the model.
- This is much easier to do in O-O systems.

好的知识模型可以描述各种现象，包括可预防或反映错误的现象

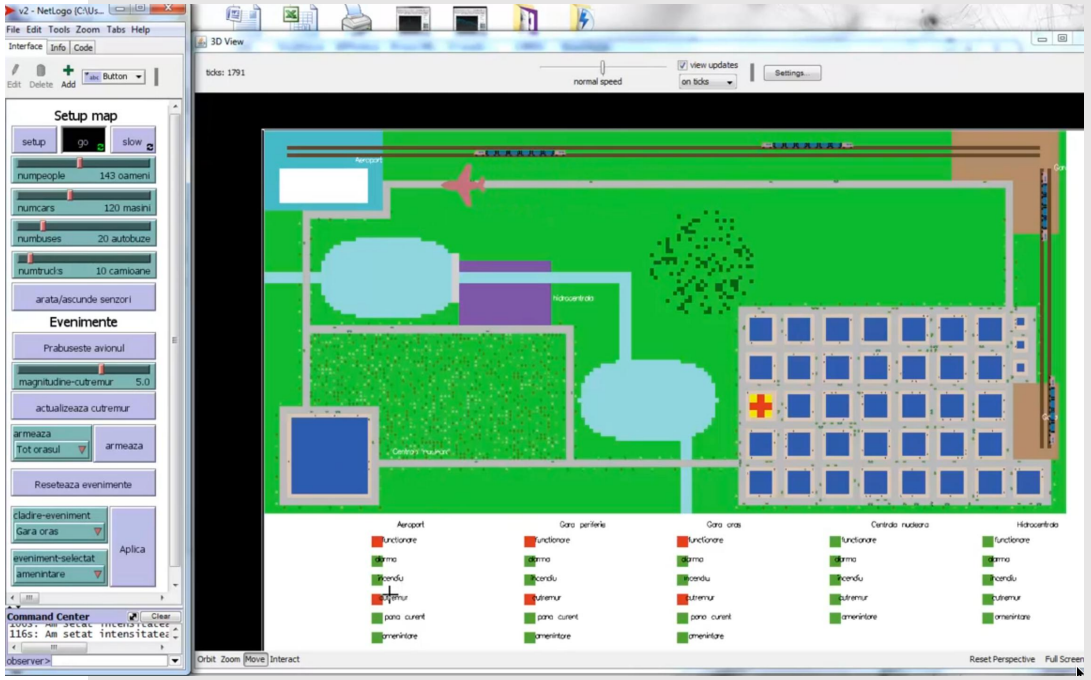
What Does a Model Give You?

- “Allocate resources efficiently”—by predicting how the system will respond to resource allocation
- “Monitor performance”—by predicting performance and tracking variances. Additionally by calibrating system performance over time.
- “Respond to unusual time-critical demands”—by showing you what you can do.
- “Handle failures”—by detecting failures early and predicting system behaviour in failure mode.

例如：交通系统的建模

A Traffic Control System

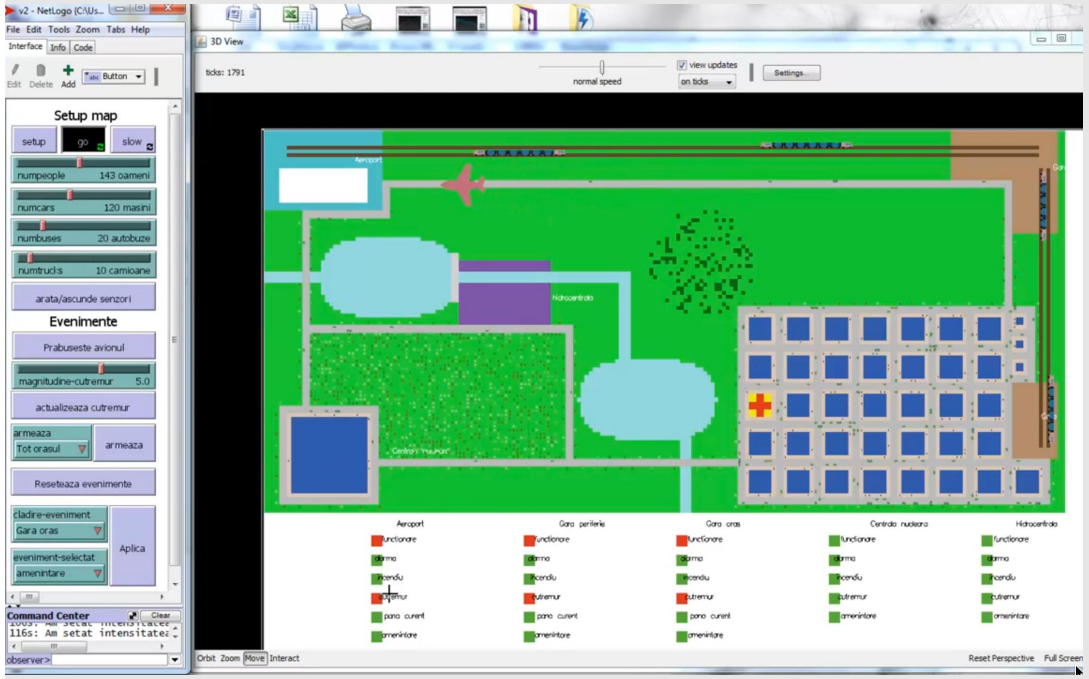
- A signal control system should:
  1. Adaptively set light intervals.
  2. Monitor traffic loads and waiting queue lengths.
  3. Allow manual or remote control of the signals.
  4. Detect signal failures, traffic overloads, unusual traffic conditions (e.g., icing, snow), and blockage of the intersection.



计算系统的功能是：低价地验证不同可能

Alternatives

- Model the intersection off line and program the traffic control system based on that model. Problems include:
  - Validation
  - Calibration
  - Evolution of traffic loads
- Use a model in an object-oriented traffic control system to predict performance and support error correction/calibration.
- Guess which is more robust?



对象 （Objects）的定义与范畴论的关系

Objects

- O-O systems are made up of objects.
- An ‘object’ is anything with the characteristics of:
  - Identity (a name)
  - Attributes
  - Behaviour
- Rows in a relational database table are *not* objects because they lack identity.
- Attributes are usually implemented as member fields in Java classes.
- Behaviour is usually implemented as member methods in Java classes.

方法（Methods）联系对象的有向线条

Methods

- A method is a portion of the behaviour of a class or class instance.
- Public methods represent class or instance behaviour that responds to messages from other classes or objects.
- A collection of public methods may make up an interface.
- The behaviour may be documented in a ‘contract’.

消息包（Message）有向线条上承载的信息

Message-Passing

- *Conceptually* (but not always) in object-oriented programming, parameters are passed to a function call as a message.
- *Ideally*, a message is an object, and it is sent to a 'port' on the destination object that is specified as an interface.
- In Java or C++, the syntax would be:

  o.foo(msg)

  Where o is the destination object, foo is a method in the interface (the port), and msg is the message.

以终为始的设计决策顺序

My Approach to Object-Oriented Design

- Define the *subsystems* and *interfaces* first.
  - In Java, subsystems are often (but not necessarily) packages.
  - Interfaces in Java are *more fundamental* than inheritance. Define interfaces first and consider inheritance during detailed design.
- Decompose your problem into interacting subsystems. Having a single main process that does everything is not good O-O design.
- Each class should have one primary function. Avoid doing too much; instead delegate responsibilities to associated classes.
- Don't dive to the bottom; define high-level subsystems before low-level subsystems.

模块化设计原则

Modularity Defined

Meyer (1988) examined this formally. He identified *criteria for software modularity* and *principles of good modular design* (next). Meyer's criteria for a good design method are that it should support:

1. modular decomposability—A design method should help with the decomposition of a novel problem into independent subproblems.
2. modular composability—A design method should favor the production of *components*.
3. modular understandability—A design method should produce modules that can be separately understood.
4. modular continuity—A method satisfies this if local changes do not propagate through the entire design.
5. modular protection—A method meets this if a minor exception condition does not propagate through the entire system.

Bertrand Meyer的设计规范

Meyer's (1988) Five Principles of Good Modular Design

These describe the features of a language that will encourage good modular design:

1. **linguistic modular units**—Modules should correspond well to the syntactic units in the language.
2. **few interfaces**—Every module should naturally communicate with as few others as possible.
3. **small interfaces**—If any two modules communicate, they should exchange as little information as possible.
4. **explicit interfaces**—When two modules communicate, it should be obvious from the text of at least one.
5. **information hiding**—All information about a module should be private by default.

https://www.computer.org/cms/Awards/images/medium/bertrandmeyer.jpg

Bertrand Meyer的设计合约

Design by Contract

Meyer designed Eiffel to implement 'design by contract'. This originally mean that a design should satisfy two rules (*don't bother learning*):

1. There are only two ways a routine call may terminate: either it fulfils its contract or it fails to fulfil it.
2. If a routine fails to fulfil its contract, the current execution of the caller also fails to fulfil its own contract.

By the second edition of Meyer's book, this had evolved to a requirement that any method must publicly specify the contract for how it is called. This includes the preconditions that must hold whenever the routine is called and the post-conditions that it guarantees when it returns. A routine should check that its preconditions are satisfied, and before completing it should check that its post-conditions are met. *(Learn.)*

In Java, a routine 'throws an exception' to indicate that it cannot fulfil its contract (but you know that already…).

设计合约的上下限(Strong / Weak)

Weak and Strong Design by Contract

- Note there are weak and strong versions of ‘design by contract’.
  - The weak version is that a method that cannot fulfil its contract must leave the system operable.
  - The strong version is that a method that cannot fulfil its contract must return the system to the state it was in when the method was called. This allows the system to try other approaches.
- To determine whether a system supports weak or strong design by contract, you examine the code to see where exceptions may be thrown:
  - Operations creating and manipulating reference types may throw.
  - Operations with primitive types generally don’t throw.

提炼系统本质的心智活动：抽象

Abstraction

Abstraction is usually viewed as an information wall in an object-oriented language that prevents the programmer from viewing the private contents of data objects. It goes further. There are four principles of abstraction (from Pratt and Zelkowitz, 1996):

- **Specialization**—This is the most common form of inheritance, where the derived object has more precise properties than the base object. The inverse concept is **generalization**.
- **Decomposition**—This is the principle of separating an abstraction into its components. The inverse concept is **aggregation**.
- **Instantiation**—This is the process of creating instances of a class. Essentially a copy operation. The inverse concept is **classification**.
- **Individualization**—The differentiation of objects for specific functions. An object of a given class has a given purpose or role. The inverse concept is **grouping** similar objects based on common purposes and roles.

面向对象是一种压缩信息的方法

Object-Oriented Design

- An object-oriented system solves a design problem using a collection of peer subsystems communicating via interfaces.
  - Subsystems may be composite, with multiple classes or objects, or atomic, consisting of a single class or object.
  - 'Peer' means that there is no single subsystem in charge.

关键词汇 （面向对象语言体系的 “本体”）

Terminology

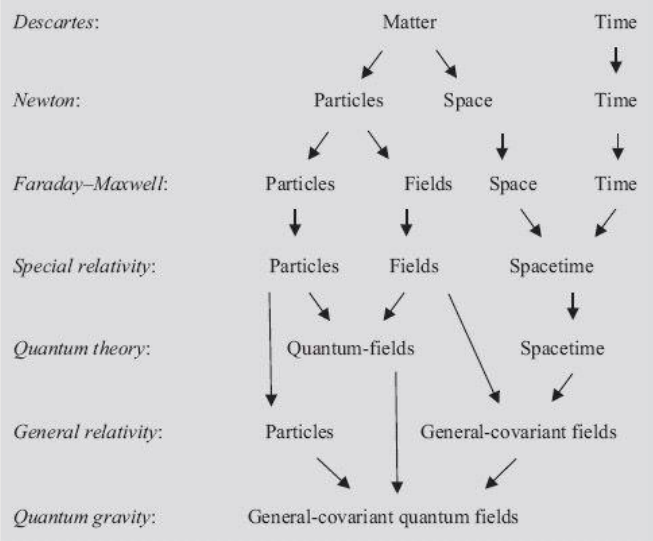
- Object Oriented Design
- Modularity
- Design by Contract
- Abstraction

总结：

- 面向对象设计(OOD)提供了一套对动态关系建模的方法和工具
  - 对象，是一种承载知识内容的本体，也是一种数据类型。
  - 方法:（Method）是一种描述对象关系的函数
- 范畴论可以支撑面向对象的建模方法：
  - 特别是：
    - 信息的模块化（Modularity）
    - 设计合约
      - 逻辑模型 / 函数输入与输出的条件
- 知识图谱的抽象方法（Ontological Engineering）
  - 除了MediaWiki等工具，Object-Oriented Design也已被大规模地运用

本周作业: (知识图谱建模)

- 根据Rovelli的时空知识图谱，用OOD的符号系统画出一个知识图谱
  - 除了图表的绘制，还要有文字内容，都直接写在Toyhouse.cc 的 Wiki 上
  - 每个小组至少要在维基上建立10个新辞条：人物、机构、科技（对象）



Covariant Quantum Gravity: An Elementary Introduction of Quantum Gravity Theory and Spinfoam Theory by Rovelli aLoop nd Vidotto

参考文献：

- An introduction to Category Theory for Software Engineers, Steve Easterbrook, U of Toronto: http://www.cs.toronto.edu/~sme/presentations/cat101.pdf
- Conceptual Mathematics, F. William Lawvere, SUNY Buffalo, http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521719162