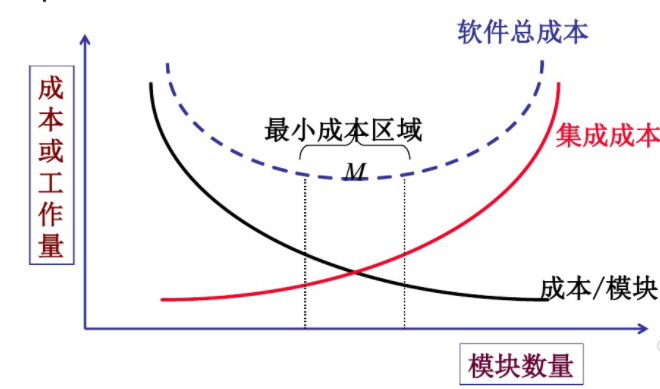
**System Analysis and Design**

**Homework Assignment 5 (lecture 12-16)**

1. **Software is divided into separately named and addressable components called modules. What is the idea of Modularity in software design? How can we design good software modules?**（莫振威）

模块化：把程序划分成若干个模块，每个模块完成一个子功能，把这些模块集总起来组成一个整体，可以完成指定的功能，从而解决问题。模块化以分治策略为依据，可以降低软件的复杂性，使软件可控、可维护、可扩展。

怎样设计好的软件模块：



1. 分解：确定模块划分的最小成本区，并保证模块的最佳性能
2. 模块完成独立的功能，满足高内聚、低耦合的设计原则：即一个模块内部各成分之间相互关联的程度尽可能高，模块间相互依赖程度尽可能低。
3. 信息隐藏：模块所包含的信息，不允许其他不需要这些信息的模块访问
4. **SOLID stands for 5 OOD principles. What are they? What ideas on OOD do they each reveal?**（胡洁珏）

**(1)What are they?**

S：single-responsibility principle（单一职责原则）

O：Open-closed principle（开放封闭原则）

L：Liskov substitution principle（Liskov 替换原则）

I：Interface segregation principle（接口分离原则）

D：Dependency Inversion Principle（依赖逆置原则）

**(2)What ideas on OOD do they each reveal?**

S–Single-responsibility principle: A class should have only one job–Modularity, high cohesion, low coupling.

（一个类应该只有一个作业——模块化性、高内聚性、低耦合性）

O–Open-closed principle: Objects or entities should be open for extension, but closed for modification.

（对扩展开放，对修改封闭）

L - Liskov substitution principle: Every subclass/derived class should be substitutable for their base/parent class.Let q(x) be a property provable about objects of x of type T. Then q(y) should be provable for objects y of type S where S is a subtype of T.

（所有子类型应该兼容夫类型。在继承关系种，父类定义的属性和方法若在子类中使用，应该具有相同的语义解释）

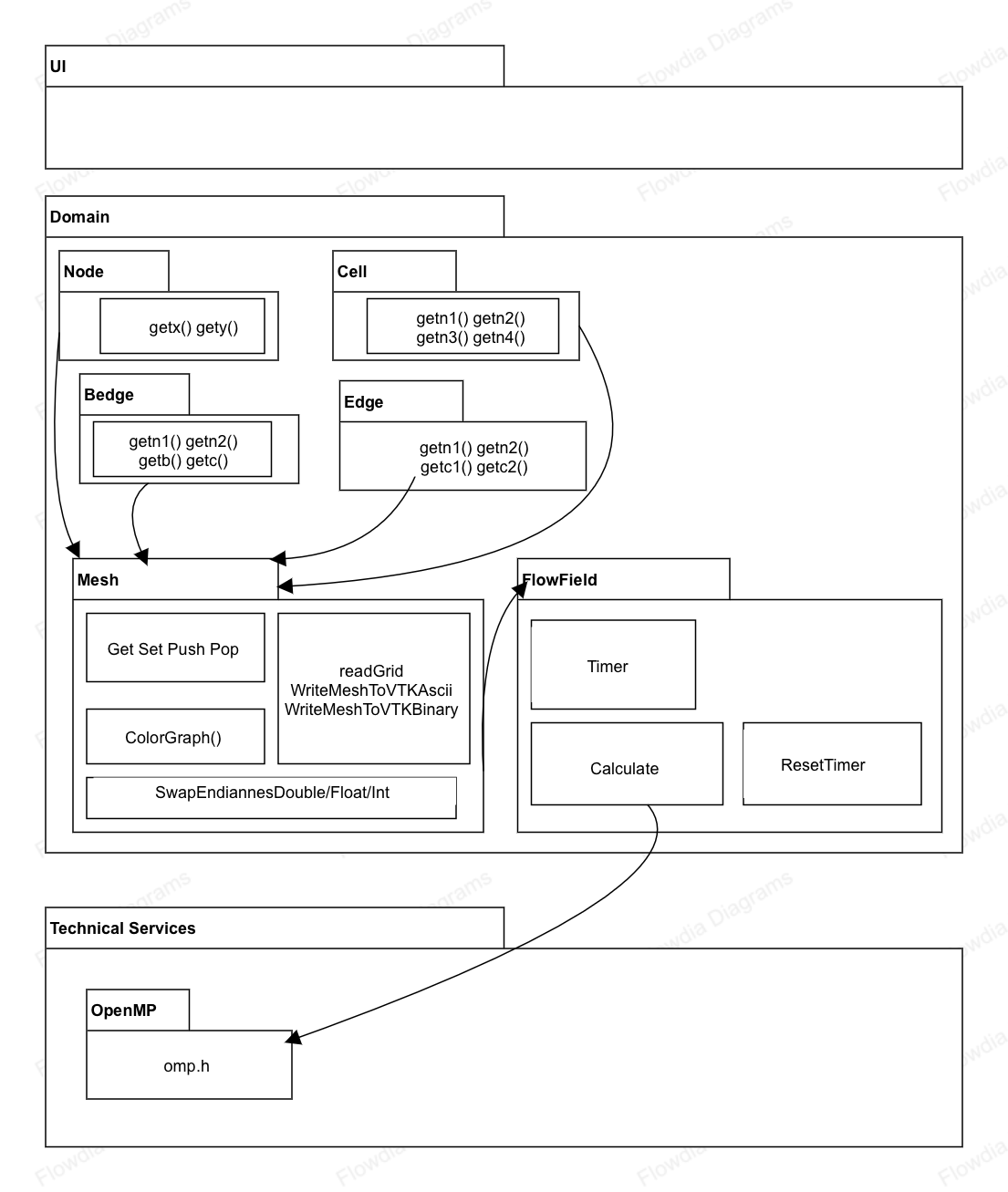
I - Interface segregation principle: A client should never be forced to implement an interface that it doesn’t use or clients shouldn’t be forced to depend on methods they do not use.

（用户用的方法应该从接口中分离出去。无用的依赖应该从接口中分离出去）

D - Dependency Inversion Principle: Entities must depend on abstractions not on concretions. It states that the high level module must not depend on the low level module, but they should depend on abstractions.

（对象不能依赖于底层的具体实现，而应该依赖于高层抽象。要依赖于抽象而不要依赖于具体类）

1. **What kind of UML diagrams are suitable for architectural modeling? Please draw a UML diagram to show the architecture of your course project.**（莫振威）

UML包图：

1. **There are two kinds of object models in object design in an iterative and evolutionary development with agile method. What are they? What are the differences between the two?**（杨宗恺）

**采用敏捷方法进行迭代和进化开发的对象设计中的两种对象模型是什么？**

答：动态对象模型和静态对象模型。

**两者有什么区别？**

答：动态模型有助于设计逻辑、代码行为或方法体，例如UML交互图（顺序图或通信图）。动态模型倾向于创建更为有益、困难和重要的图形。静态模型有助于设计包、类名、属性和方法特征标记（但不是方法体）的定义，例如UML类图。

静态和动态建模之间具有关系，敏捷建模对此的实践是并行创建模型：花费较短的时间创建交互图（动态），然后转到对应的类图（静态），交替进行。

UML初学者一般会认为静态视图的类图是重要图形，但事实上，大部分具有挑战性、有益和有效的设计工作都会在绘制UML动态视图的交互图的时候发生。需要哪些对象，它们如何通消息和方法进行协作，通过动态对象建模（例如绘制顺序图）才能真正落实这些准确和详细的结论。

应该把时间花费在交互图（顺序图或通信图），而不仅是类图上。

忽视这一准则是十分常见的UML错误实践。

先进行动态对象建模，然后进行静态对象建模

如果开发人员正在应用并行创建多个模型的敏捷建模实践，那么他们应该同时绘制交互图和类图。

1. **Why the object design skill is much more valuable than knowing UML notation?**

（陈家龙）

答：这是因为UML图只是面向对象设计的一种表示形式。在管理复杂的，大型的项目时，通常需要先对业务逻辑中所涉及到的对象之间进行建模，而且在建模过程中需要满足一定的面相对象设计原则，比如单一职责原则，开闭原则等等。在进行面向对象建模时用到的这些原则和技巧会直接决定对象之间的关系，进而影响了整个项目的可维护性，可拓展性等等性能。

而UML图只是对象之间的关系的一种可视化表示，通过UML图可以把面向对象的设计结果清晰的表现出来，UML图只能反应出面向对象设计，而不能影响面向对象设计，即使没有UML，面向对象设计也有可能通过其他表现形式反映。

因此，学习面向对象设计的技巧时我们考虑的是如何利用它来实现对整个项目的最优化的建模，而学习UML时只是考虑如何准确的把这个建模反应出来，显然前者比后者更重要，更有价值。

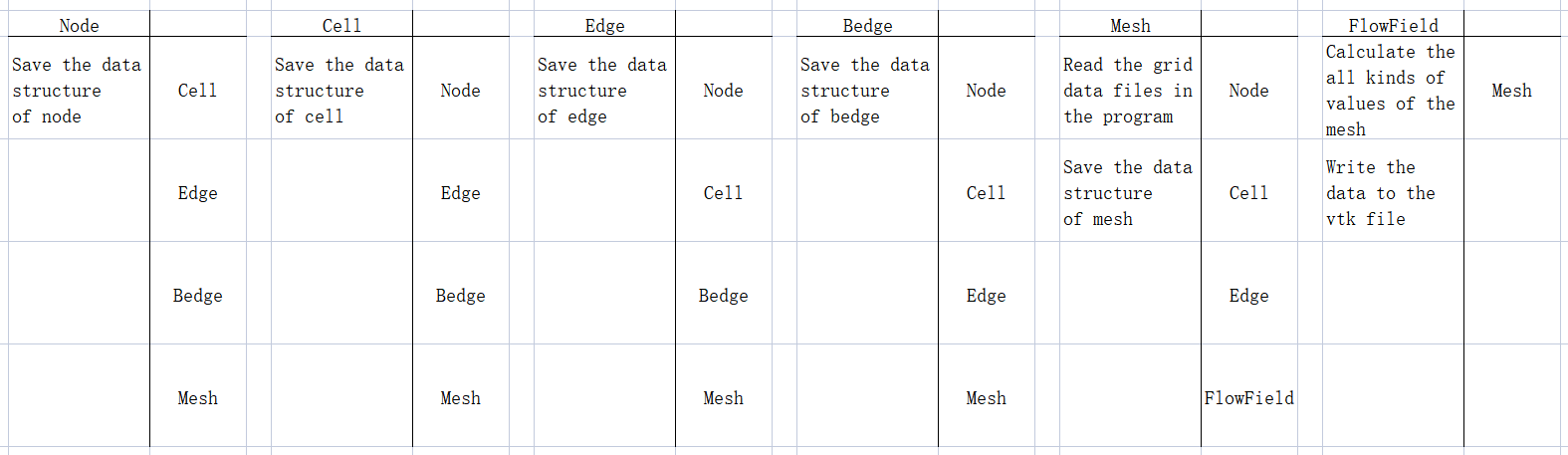
1. **What are the required knowledge for Fundamental Object Design?** （胡若晴）

答：职责分配原则和设计模式。

1. **A guideline for dynamic object modeling states that “Spend time doing dynamic modeling with interaction diagrams, not just static object modeling with class diagrams”. What does this statement mean?**（丁维力）

答：在考虑真正的OO设计细节时，就必须要“落实”发送哪些消息、发送给谁、以何种顺序发送等具体信息。UML使用交互图来描述对象间通过消息的交互，交互图可以用于动态对象建模。UML用类图表示类、接口及其关联，类图用于静态对象建模。所以要花时间使用交互图进行动态建模，而不仅仅是使用类图进行静态对象建模。花足够的时间做动态建模，可以更深入的理解对象及其行为。

1. **Find possible software objects (possible classes you think important) for mesh applications and draw CRC cards for each object.**（杨宗恺）

****

1. **Explain (or summarize) in your own words on how to do Dynamic Object modeling.**（崔子潇）

答：动态对象建模是指设计一个类的行为、方法和逻辑，例如UML交互图（顺序图或通信图）。它需要确定某一事件中有哪些对象，对象之间要怎样通过消息或方法进行协作，需要将这些内容整理成UML交互图的形式来详细地确定它们的关系。

1. **Explain (or summarize) in your own words on how to do Static object modeling.**

（崔子潇）

答：静态对象建模是指需要设计一个包、类、属性或方法特性，例如UML类图。做这个建模需要确定好对象所属的包，确定类的名称，并写好类的属性和操作集，最后整理成UML类图。

1. **Revise the collection of the following artifacts you have done for your course project.**

* Glossary
* Supplementary Specification
* SSD
* System Operation Contracts
* Domain Model (Conceptual class diagrams)
* System Architecture
* Interaction diagram for dynamic object modeling
* Domain class diagram

Notice that some sections may be not complete in your document at this moment.

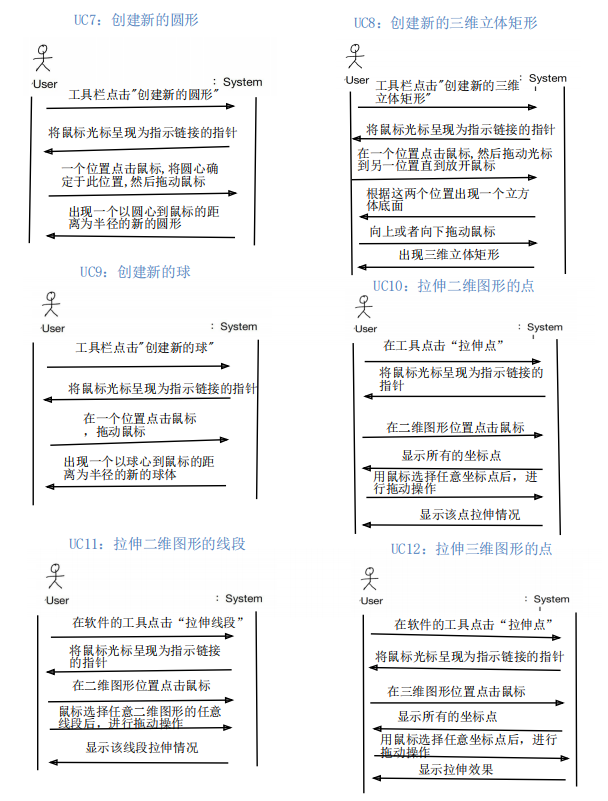
**SSD系统顺序图**

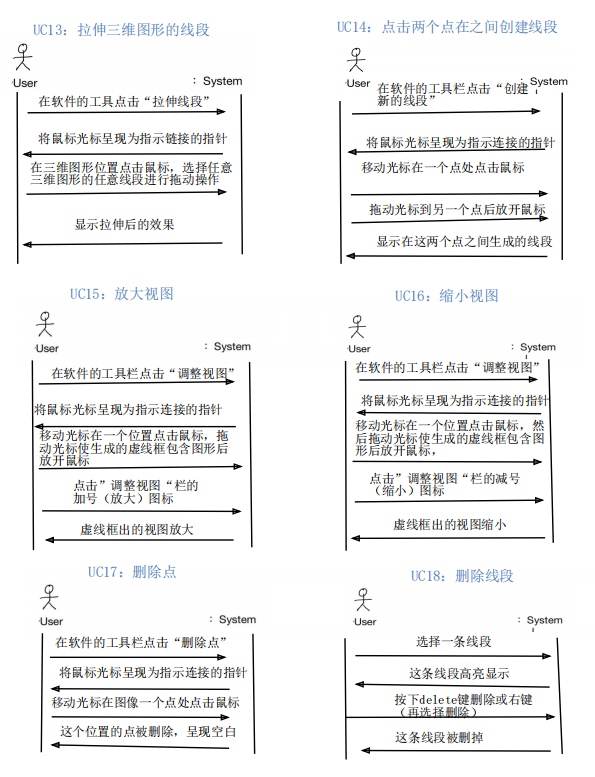
**修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 系统顺序图1.0 | 2021年6月13日 | 创建系统顺序图 | 胡洁珏 |

**图示：**





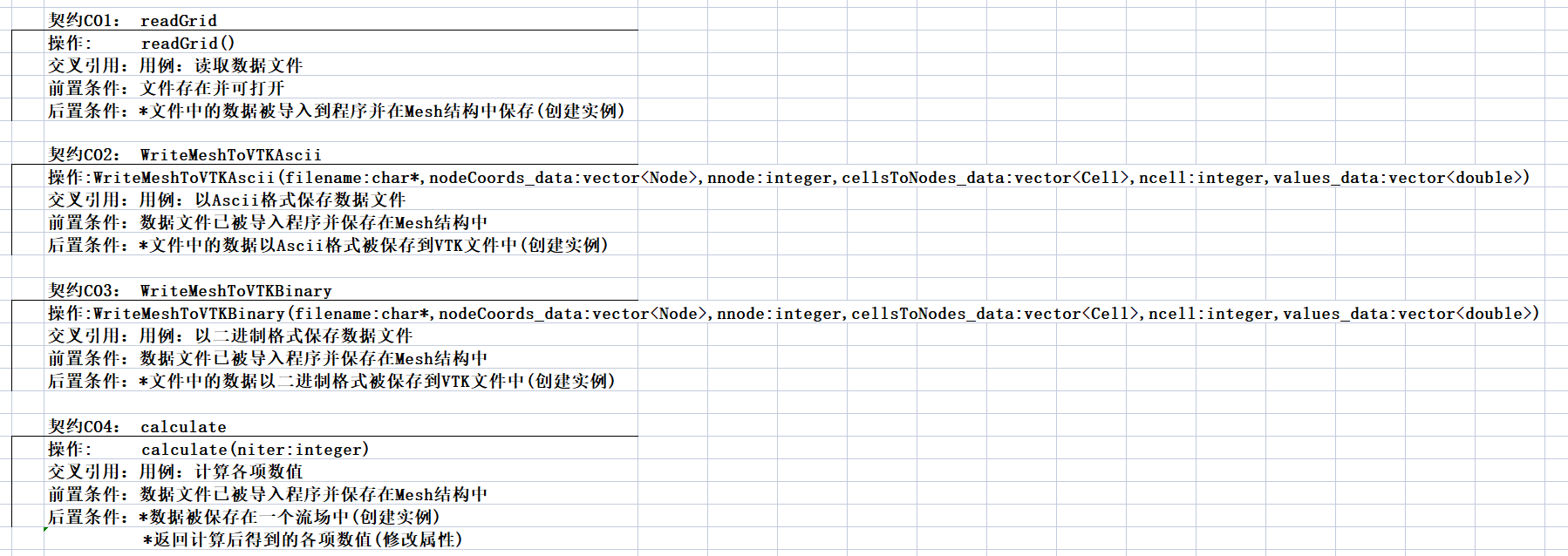


**System Operation Contracts 操作契约**

**修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 操作契约1.0 | 2021年6月13日 | 创建操作契约 | 胡若晴 |

**图示：**

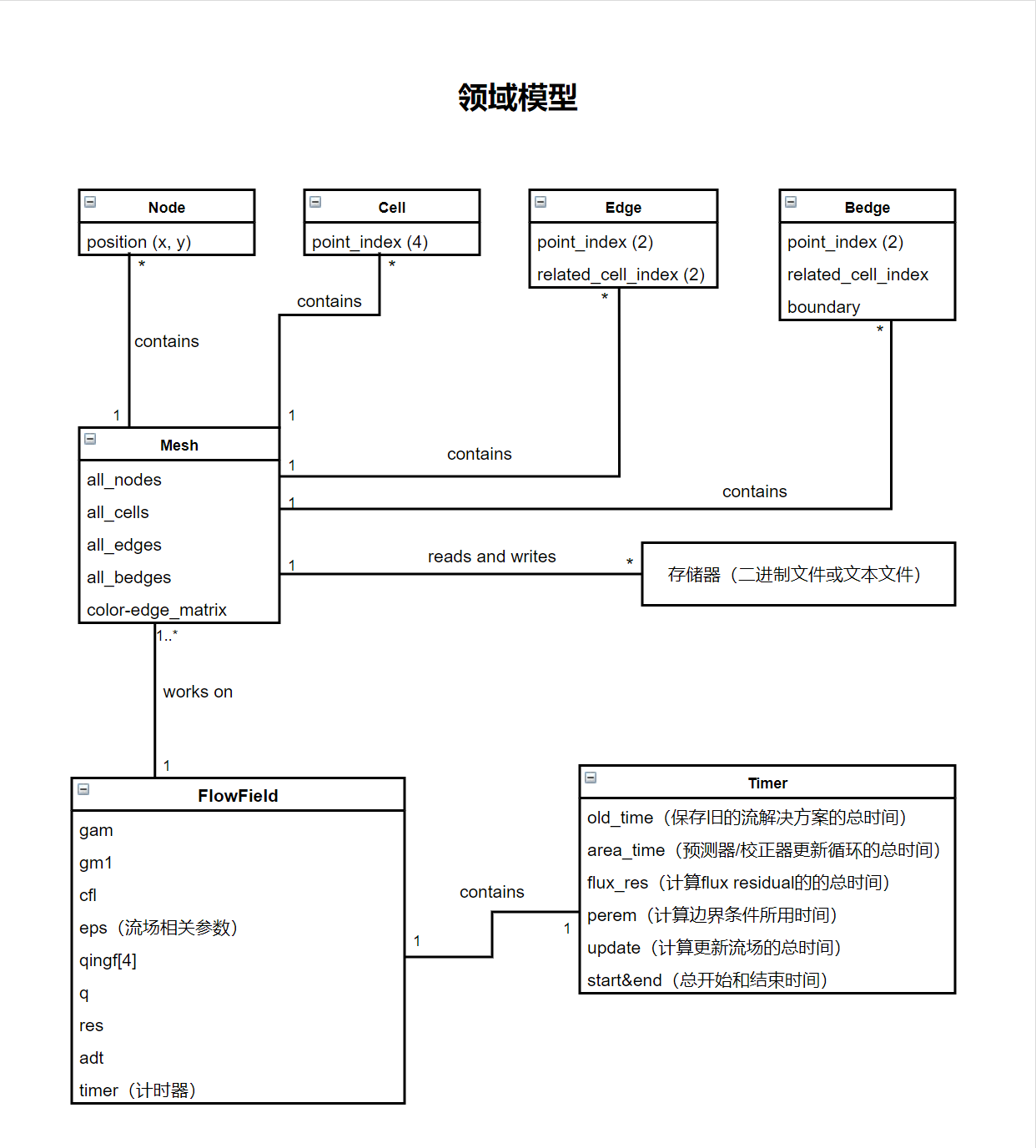


**Conceptual class diagrams 概念类图**

**修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 概念类图1.0 | 2021年5月22日 | 创建概念类图 | 杨宗恺 |
| 概念类图1.1 | 2021年6月13日 | 修改概念类图 | 崔子潇 |

**图示：**

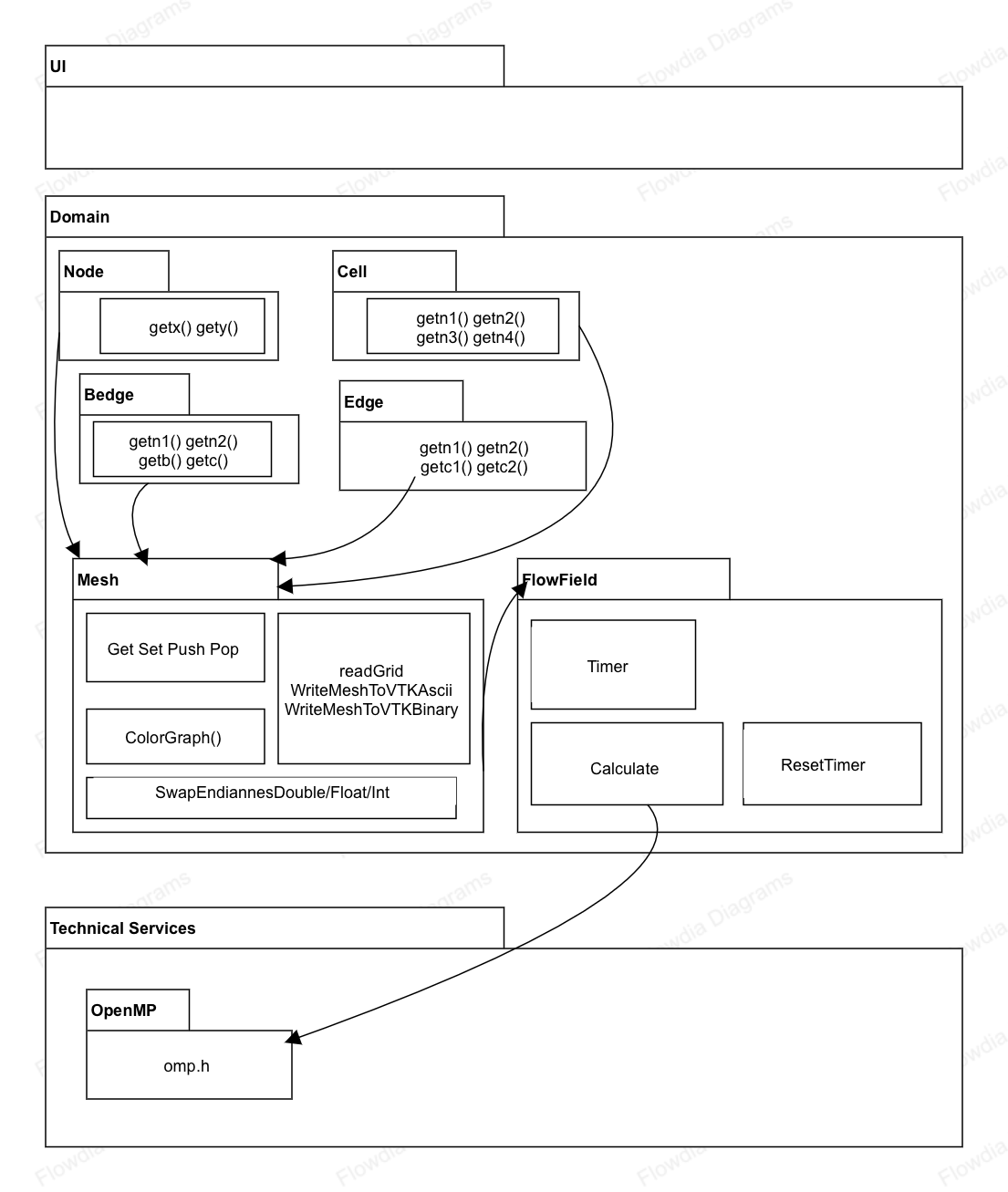


**System Architecture 系统架构**

**修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 系统架构图1.0 | 2021年6月13日 | 创建系统架构图 | 莫振威 |

**图示：**



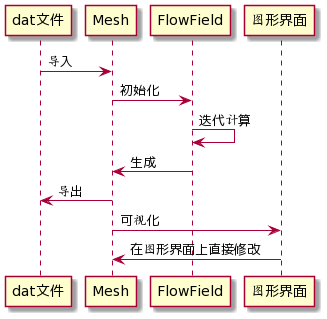
**Interaction diagram for dynamic object modeling**

**动态对象建模的交互图**

**修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 动态交互图1.0 | 2021年6月13日 | 创建动态交互图 | 陈家龙 |

**图示：**



**Domain class diagram 领域类图**

**修订历史**

|  |  |  |  |
| --- | --- | --- | --- |
| **版本** | **日期** | **说明** | **作者** |
| 领域类图1.0 | 2021年6月13日 | 创建领域类图 | 杨宗恺 |

**图示：**

