设计原则：

1. 模块化：进程管理模块分为重生点管理、存档管理、重生状态管理三个模块，重生点管理与重生状态管理的耦合度较高，存在数据耦合，与存档管理的耦合度与内聚度较低，与重生点管理与重生状态管理放在一个模块是巧合内聚，只是出于方便才放在同一个模块中。
2. 接口：进程管理模块规格说明：目标：保证游戏流程的正常进行，管理角色死亡或是退出游戏后接下来的重生点及状态。前置条件：游戏操作方面及画面演出方面正常，面对角色死亡与退出游戏时发挥作用。协议：结构简单采用直接授权。后置条件：完成进程管理之后再回到操作管理模块继续游戏。质量属性：不会因为角色的重生或是游戏的退出而使流程出错。
3. 信息隐蔽：重生管理模块与存档管理模块之间并没有信息交流，信息隐蔽程度良好。
4. 增量式开发：单纯的进程管理模块实现按顺序开发即可，结合其他模块考虑增量式开发顺序表。
5. 抽象：进程管理模块抽象为对不同情况的玩家做出不同的地点、状态安排，情况分角色死亡与退出游戏。
6. 通用性：将特定的上下文环境信息参数化，去除前置条件，简化后置条件。

面向对象的设计：

1. 术语：根据对象设计了对象的类和接口，游戏使用的基础框架为一个抽象的类，后面的完整模块继承了类的使用，关于这方面主要涉及到的类是重生点类与存档类，道具类可能会对重生状态造成影响。

2. 继承与对象组合：在进程管理模块涉及的类中并没有涉及相关知识，仅作为补充，父类为游戏的基础框架和内核单元设计，后续开发过程使用了子类继承父类的设计思想，在父类的基础上进行子类的功能开发，本质上是一种完善和补充。

1. 可替换性：利斯科夫替换原则的主要用途是确定在什么时候一个对象可以安全地被另一个对象所替代。如果我们在设计新类时遵循这个原则，我们就可以在不修改现存代码的情况下使用新的子类。
2. 德米特法则:一般情况下，遵循德米特法则的设计具有更少的依赖关系，而类之间的依赖关系越少，软件故障也就越少，软件也就越易于修改。
3. 依赖倒置:可以把两个类之间的依赖链接方向进行倒置，本技术可以用来消除类形成的依赖循环，本模块中并不设计依赖循环，故不用使用该技术。