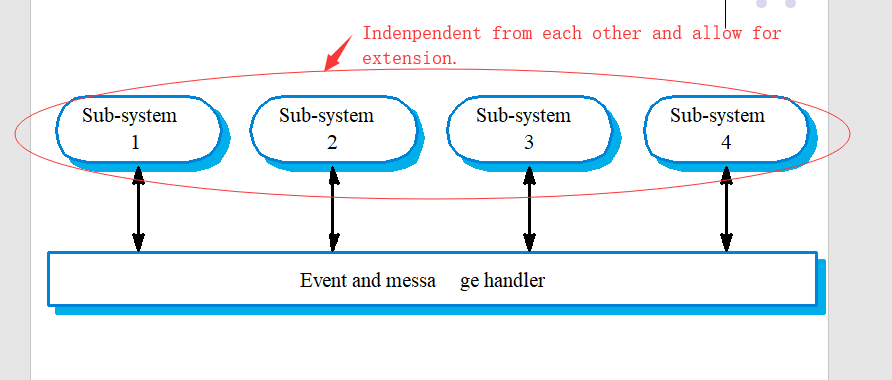
The event-driven architecture have a central event handler for receiving event when a event occur , the handler should take the event message and broadcast it to all its sub system , it is up to the sub-system to decide whether it can handle this event. One event can be handled by multiple sub-system concurrently. This kind of architecture should have following characteristic , functional decoupling, maintainable, extendable, robust and reusable.



Following [Single Responsibility Principle](http://blog.csdn.net/vking_wang/article/details/8450886) to achieve functional decoupling, each sub-system should only designed for handling one type of event. So that changes to one type event handling does not affect other. This kind of design also has a benefit when one sub-system is down, the remain system can still be running.

Every sub-system should regard same from the view of event handler, so that it would inform every sub-system when a event occur, [Liskov Substitution Principle](http://blog.csdn.net/vking_wang/article/details/8455621) can be applied to achieve this purpose.

Applying [Interface Segregation Principle](http://blog.csdn.net/vking_wang/article/details/8455631) to the interface between event handler and sub-system to remove the dependency between event handler class and sub-system implementation.