1. Introductions

The increasing and various requirements make software with complex and monolithic architecture. Moreover, to meet the fast change in the business domain, it is necessary for vendors and developers to accelerate the development and modification of software. In this scenario, the structured design may make high cost and difficulty for the maintenance and extension of software. The object-oriented programming (OOP) technology first introduced in Simula 67 [1] get more and more emphasis. Object-oriented programming emphasizes object aspect and supports programmers to develop reusable programs and objects. The reusable programs and objects increase the efficiency of software development and reduce the cost of maintenance. Many programming languages, such as C++, Objective-C and Java, have been developed to assist programmers in object-oriented programming.

Workflow is probably one of the most exciting areas of research that has emerged in the past few years. The concepts and ideas have been around in one form or another for a long time: computer supported cooperative work, form processing, cooperative systems, office automation [5]. Workflow Management Systems (WfMSs) are used to execute workflow process, the computerized model of business tasks. Although the concepts of WfMS have been full-blown, there are still many issues for the design and technology used for the development of WfMSs.

In computer science and software engineering, *reusability* is the likelihood a segment of structured code can be used again to add new functionalities with slight or no modification [48] . Subroutines or functions are the simplest form of reuse. Proponets claim that objects and software components offer a more advanced form of reusability.

In this paper, we purpose an object-oriented model for workflow management system. In this model, we emphasize the “reusability” of components of WfMS. According to [7][8], many workflow systems are built using object-oriented programming language like C++ or Java. However, they do not fully exploit the potential of object technology. Rather, they regard it mainly as an implementation technique [8]. Object-oriented programming languages permit the reuse of objects. But, can the object-oriented programming languages provide reusability for all the components in a WfMS? To answer this question, we will first clarify and define the “reusability” of components in WfMS and then point out the difficulties that the programmers may encounter when using object-oriented programming language to implement the components with reusability we defined. Finally, we will introduce our model and how we overcome these difficulties.

1.1 The Reusability of Object-Oriented Programming Language

In this section, we introduce how object-oriented programming language support reusability. In 1987, Dr. Brad Cox, a great master of object-oriented programming introduced the “software IC” concept [4]; he hopes that the development of software can like the hardware development to use existing components to generate a new component. To make the software IC possible, the reusability of classes or components is very important. Below, we introduce the features of object-oriented languages that support the reuse of classes.

- **Encapsulation:** *Encapsulation* is used for *information hiding*. Information hiding is leaving out some details of implementation on purpose from the public. The motivation of information hiding is to reduce dependency between classes. A

class can store and retrieve the data encapsulated in another cloass via methods (e.g. setter and getter), or if not needed, no mechanism to access hidden code or data is provided. Because of low dependency, the change of a class may have less or none influence on other classes. Therefore, the encapsulation feature promotes the reusability by reducing dependency between classes.

- **Inheritance**: an *inheritance* is a way to form new classeses or objects using pre-defined objects or classes where new ones simply take over old ones's implemetions and characterstics. It is intended to help reuse of existing code with little or no modification. Many object-oriented programming languages permit a class or object to replace the implementation of an aspect—typically a behavior—that it has inherited. This process is usually called *overriding*.
- **Polymorphism**: *polymorphism* refers to the ability for references and collections to hold objects of different types, and for the methods invoked on those references to find the right type-specific behavior, with the correct behaviour being determined at run time (this is called *late binding* or *dynamic binding*). Polymorphism allows programmers to use the same name to invoke different operations on objects of different data types and promotes the reusability of objects.