**Особенности интеграционного тестирования для объектно-ориентированного программирования**

Программный проект, написанный в соответствии с объектно-ориентированным подходом, будет иметь ГМП, существенно отличающийся от ГМП традиционной "процедурной" программы. Сама разработка проекта строится по другому принципу - от определения классов, используемых в программе, построения дерева классов к реализации кода проекта. При правильном использовании классов, точно отражающих прикладную область приложения, этот метод дает более короткие, понятные и легко контролируемые программы.

Объектно-ориентированное программное обеспечение является событийно управляемым. Передача управления внутри программы осуществляется не только путем явного указания последовательности обращений одних функций программы к другим, но и путем генерации сообщений различным объектам, разбора сообщений соответствующим обработчиком и передача их объектам, для которых данные сообщения предназначены. Рассмотренная ГМП в данном случае становится неприменимой. Эта модель, как минимум, требует адаптации к требованиям, вводимым объектно-ориентированным подходом к написанию программного обеспечения. При этом происходит переход от модели, описывающей структуру программы, к модели, описывающей поведение программы, что для тестирования можно классифицировать как положительное свойство данного перехода. Отрицательным аспектом совершаемого перехода для применения рассмотренных ранее моделей является потеря заданных в явном виде связей между модулями программы.

Перед тем как приступить к описанию графовой модели объектно-ориентированной программы, остановимся отдельно на одном существенном аспекте разработки программного обеспечения на языке объектно-ориентированного программирования (ООП), например, C++ или С#. Разработка программного обеспечения высокого качества для MS Windows или любой другой операционной системы, использующей стандарт "look and feel", с применением только вновь созданных классов практически невозможна. Программист должен будет затратить массу времени на решение стандартных задач по созданию пользовательского интерфейса. Чтобы избежать работы над давно решенными вопросами, во всех современных компиляторах предусмотрены специальные библиотеки классов. Такие библиотеки включают в себя практически весь программный интерфейс операционной системы и позволяют задействовать при программировании средства более высокого уровня, чем просто вызовы функций. Базовые конструкции и классы могут быть переиспользованы при разработке нового программного проекта. За счет этого значительно сокращается время разработки приложений. В качестве примера подобной системы можно привести библиотеку Microsoft Foundation Class для компилятора MS Visual C++ ".

Работа по тестированию приложения не должна включать в себя проверку работоспособности элементов библиотек, ставших фактически промышленным стандартом для разработки программного обеспечения, а только проверку кода, написанного непосредственно разработчиком программного проекта. Тестирование объектно-ориентированной программы должно включать те же уровни, что и тестирование процедурной программы - модульное, интеграционное и системное. Внутри класса отдельно взятые методы имеют императивный характер исполнения. Все языки ООП возвращают контроль вызывающему объекту, когда сообщение обработано. Поэтому каждый метод (функция - член класса) должен пройти традиционное модульное тестирование по выбранному критерию C (как правило, С1 ). В соответствии с введенными выше обозначениями, назовем метод Modi, а сложность тестирования - V(Modi,C). Все результаты, полученные в лекции 5 для тестирования модулей, безусловно, подходят для тестирования методов классов. Каждый класс должен быть рассмотрен и как субъект интеграционного тестирования. Интеграция для всех методов класса проводится с использованием инкрементальной стратегии снизу вверх. При этом мы можем переиспользовать тесты для классов-родителей тестируемого класса , что следует из принципа наследования - от базовых классов, не имеющих родителей, к самым верхним уровням классов.