МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Гжельский государственный университет»** (ГГУ)

Колледж ГГУ

Специальность 09.02.07 Информационные системы и программирование

**Реферат**

**По предмету «Моделирование и анализ программного обеспечения»**

**На тему «Обратное проектирование»**

ВЫПОЛНИЛ:

Студент группы ИСП-0-17

Филипович А.А.

ПРОВЕРИЛА:

Прокуронова А.Ю.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

п. Электроизолятор

2019 г.

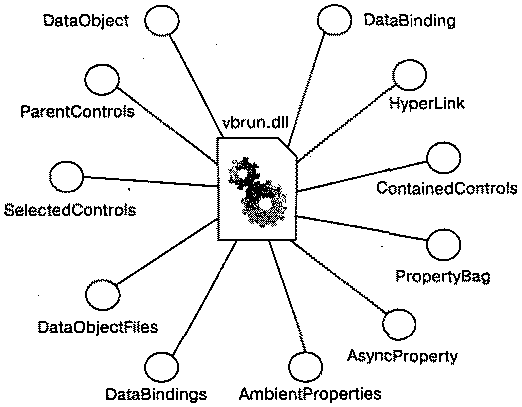
***Обратное проектирование***

*Обратным проектированием* (Reverse engineering) называется процесс преобразования в модель кода, записанного на каком-либо языке программирования. В результате этого процесса вы получаете огромный объем информации, часть которой находится на более низком уровне детализации, чем необходимо для построения полезных моделей. В то же время обратное проектирование никогда не бывает полным. Как уже упоминалось, прямое проектирование ведет к потере информации, так что полностью восстановить модель на основе кода не удастся, если только инструментальные средства не включали в комментариях к исходному тексту информацию, выходящую за пределы семантики языка реализации.  
  
 Обратное проектирование диаграммы классов осуществляется следующим образом:  
  
1. Идентифицируются правила для преобразования из выбранного языка реализации. Это можно сделать на уровне проекта или организации в целом.  
  
2. С помощью инструментального средства указывается код, который будет подвергнут обратному проектированию.  
  
3. Используя инструментальные средства, создаётся диаграмма классов путем опроса полученной модели. Следует начать, например, с одного или нескольких классов, а затем расширить диаграмму, следуя вдоль некоторых отношений или добавив соседние классы. При этом можно раскрыть или спрятать детали содержания диаграммы в зависимости от ваших намерений.

Обратное проектирование диаграммы компонентов осуществляется так:

1. Выберите целевое представление. Исходный код можно реконструировать в компоненты, а затем и в классы. Двоичные библиотеки можно подвергнуть обратному проектированию для раскрытия их интерфейсов. Исполняемые программы поддаются обратному проектированию в наименьшей степени.
2. С помощью инструментальных средств укажите на код, который вы хотите подвергнуть обратному проектированию. Воспользуйтесь инструментами для генерации новой модели или модификации существующей, для которой ранее было проведено прямое проектирование.
3. Воспользуйтесь инструментальными средствами для создания диаграммы компонентов путем сверки с моделью. Например, можно начать с одного или нескольких компонентов, а затем расширять диаграмму, следуя по связям или переходя к соседним компонентам. Раскройте или спрячьте детали этой диаграммы компонентов в соответствии с тем, что именно вы хотите донести до читателя.

В качестве примера на [рис. 29.6](http://alice.pnzgu.ru:8080/~zsa/sql/titan_zsa/uml_htm_gol/gl_29.htm#r29_6) представлена диаграмма компонентов, полученная в результате обратного проектирования компонента ActiveX vbrun.dll. Видно, что компонент реализует 11 интерфейсов. Имея такую диаграмму, вы начинаете понимать семантику компонента и можете переходить к исследованию деталей интерфейсов.



Чаще всего при обратном проектировании исходного кода, а иногда и двоичных библиотек или исполняемых программ, прибегают к помощи системы управления конфигурацией. Это означает, что вы будете работать с конкретными версиями файлов или библиотек, совместимых друг с другом. В таких случаях бывает полезно включить помеченное значение, представляющее версию компонента, - ее может предоставить система управления конфигурацией. Тогда вы сможете воспользоваться UML для визуализации истории компонента при смене версий.