Моделирование и управление

Цель этапа – выбрать модель управления взаимодействия подсистем, определиться со способом, которым будет организована работа всей системы, при этом определить типы управляющих воздействий, необходимые данные. В результате будет построена динамическая система взаимодействия системы (?). По этой модели можно будет отследить передачу потока управления системы и её подсистемами.

Поток управления *(фото 19.05.2016, 10:35)*

```
for(…) {
        for(…) {
                if(…) {
                        …
                }
        }
}
```

Граф потока управления – упрощённая структурная схема, нужен для тестирования ($C(G) = N_{дуг} - N_{вершин} + 2 = 9 - 7 + 2$)

Пути:
1)1-2-4
2)1-2-3-2-4
3)1-2-3-5-6-7-3-2-4
4)1-2-3-5-7-3-2-4

Модели управления:

1. Централизованого управления
   Центральная управляющая система, координирующая работу подсистем
2. Управление основанное на событиях
   Центральной управляющей системы нет, вся работа управляется внешними средствами. События могут происходить как внутри системы, так и быть внешними.

Централизованное управления

Модель «Вызов-Возврат»

(фото 19.05.2016, 10:50)

```
main() {
        f1();
        f2();
}

f1() {
        f11();
        f12();
}
f2() {
        f21();
        f22();
}
```

Модель используется в классических языках (вызов-возврат из функций)
+ Простота написания и чтения
– Только последовательно работающие подсистемы

Модель «Диспетчера» *(фото 19.05.2016, 10:55)*

Контроллер – центральный процесс диспетчера, который полностью контролирует работу остальных модулей. Не предполагается внешних воздействий. Момент запуска нового модуля определяется только внутренним состоянием контроллера. Контроллер диспетчера, в отличии от остальных модулей, работает постоянно.

+ Возможность разрабатывать модели с параллельно работающими подсистемами
– Нагрузка на контроллер и требования к его надёжности
– Усложнённая отладка из-за сложной архитектуры
– Проблемы с синхронизацией
– Скорость работы
– Для систем реального времени не подходит (только для мягких систем реального времени)
– Сложная обработка исключительных ситуаций.

Управление на событиях

В отличии от модели централизованного управления, где следующий этап определялся внутренним состоянием системы, т.е. набором значений её переменных состояний, в модели управления на событиях следующий этап определяется внешними воздействиями на систему. При

этом особенность заключается в том, что внешнее воздействие невозможно планировать. Обработчик внешнего воздействия проектируется таким образом, что управляющие воздействия никаким образом нельзя контролировать или предсказать.

Модель «Передачи сообщений»

События рассматриваются как сообщения, которое передаётся от одной подсистемы и может быть обработано одной из других подсистем. *(фото 19.05.2016, 11:15)* Каждая из подсистем способна обрабатывать определённые типа сообщений или запросы, при добавлении подсистемы в систему в обработчики сообщений регистрируются те сообщения, которые может принимать и обрабатывать добавляемая система. При необходимости запросить какие-то действия, подсистема отправляет какие-то сообщения, которые принимаются обработчиком сообщений. Обработчик передаёт это сообщение той подсистеме, которая может его обработать. При необходимости обработчик организует двухточечное взаимодействие между системой, отправившей сообщение, и системой, обрабатывающей его.

Подсистема, оправляющая сообщение, не знает какой подсистемой оно будет обработано, т.е. отсутствует жёсткая связь между отправителем и обработчиком.

+ Гибкость – легко добавить новые подсистемы, просто зарегистрировав их
+ Надёжность – дублирующие подсистемы
– Не гарантируется время обработки сообщения – подсистема-отправитель не знает кем будет обработано её сообщение
– Возможны конфликты – на одно сообщение отвечают две системы одновременно

Модель «Управления на прерываниях»

(фото 19.05.2016, 11:25)

Прерывание – поступление внешнего запроса и передача управления обработчику, прерывая выполнение текущего кода. Используются для обработки долгих задач (ввод-вывод) в системах реального времени, где скорость отклика на внешнее событие - это ключевое требование.

+ Скорость реакции на события
– Сложность разработки и отладки
– Ограниченное количество прерываний

Компромис на основе «Управлением на прерываниях» и «Диспетчера». Обычный ход работы управляется на основе диспетчера, а особо важные ситуации обрабатываются как прерывания.

Модульная декомпозиция

На этапе модульной декомпозиции решается задача декомпозиции и разбиения подсистем на модули. Поскольку модули меньше, чем подсистемы, применять методы разбиения аналогично с подсистемами не рационально.

Объектно-ориентированная декомпозиция

Известно из ООП. Проектирование классов, объектов…

Модель потоков данных aka «Конвейерная» модель

Основная идея заключается в том, что подсистема представляется в виде набора преобразователей (фильтров), через которые проходят данные. Эти фильтры могут выполнять некоторые преобразования данных. Они могут быть соединены последовательно или параллельно, обрабатывать данные поэлементно или целиком. *(фото 19.05.2016, 12:05)*

+ Отдельные фильтры можно использовать повторно
+ Легкая реализация как последовательного так и параллельного взаимодействия
+ Легко модифицируется простым добавлением
– Требуется единый формат данных, т.к. данные передаются по цепочке => избыточность, без неё требуется согласование данных выхода одного фильтра со входом другого
– Не подходит для интерактивных систем, ориентирующихся на работу с внешними воздействиями, т.к. невозможно прогнозировать

Основы проектирования интерфейсов пользователя

(фото 19.05.2016, 12:20)

Основные принципы проектирования интерфейсов:

1. Учёт знаний пользователя – в интерфейсе должны быть взяты термины и понятия, взятые из предыдущего опыта пользователя

2. Принцип согласованности – однотипные операции должны выполнятся одним и тем же способом
3. Минимум неожиданностей – поведение ПП должно быть прогнозируемо
4. Способы восстановления после ошибочного завершения и предотвращения нежелательных действий
5. Руководство пользователя
6. Необходимо учитывать разнородность пользователей

Способы взаимодействия интерфейсов:

1) Прямое манипулирование
+ Интуитивная понятность, простота строения
+ Скорость
– Сложная реализация
– Не для всех объектов возможно подобрать визуальный образ

2) Строка меню
+ Меньше вероятность ошибочных действий
+ Компактность
– Для опытных пользователей слишком медленно
– Если меню слишком вложенное сложно что-то найти

3) Заполнение данных форм
+ Простота
+ Удобство
– Занимает много мета на экране

4) Командный язык
+ Гибкость
+ Функциональность
– Сложность изучения
– Высокая вероятность ошибок

5) Естественный язык
+ Подходит для неопытных пользователей
– Сложность реализации

Анти-паттерны

Анти-паттерн – это отрицательный приём, примеры плохой практики проектирования ПП

1. Программирование «Copy-Paste»
   Причина – лень
   – Усложнение модификации скопированного кода
   – Снижается качество кода
   – Усложняется поддержка кода
2. «Спагетти»-код
   Причина – лень и удалённая разработка
   Слабо структурированная и плохо сконструированная система
   – Поддержка становится невозможной
3. «Золотой молоток» – один и тот же паттерн для всего
   Причина – недостаток опыта и самоуверенность
   Решение – предлагать для каждой задачи несколько решений, выбирать конкретное, сопоставляя их между собой
4. «Магические» числа
   Непонятные значения, возникшие ниоткуда
   – Непонятно что означают
5. «Hard code»
   Внесение данных об окружении в реализацию
   Причина – недостаток опыта
   – Невозможность изменить данные без компиляции
6. «Soft code»
   Излишняя параметризация
   Причина – попытка избежат п. 5

7. Ненужная сложность
   Возникает от применения сложных решений, где возможны простые
   – Сложность в сопровождении
8. «Лодочный якорь»
   Сохранение устаревшего и неиспользуемого кода
9. Изобретение «велосипеда»
10. Изобретение «велосипеда» с квадратными колёсами
11. «Поток лавы»
    Фрагмент кода, перед которым стоит комментарий // Не знаю как это работает, но мне это надо
12. Программирование перебором (программная индукция)
    Подбор решения исходя из правильного решения
13. Безумное комментирование
14. Недостаточная проверка входных данных