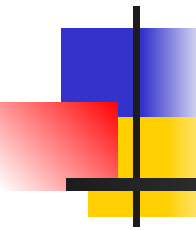
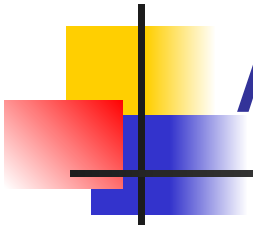


Design Patterns. Анти-паттерны

Немчинский Сергей

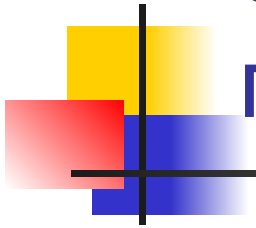
2008



Анти-паттерны

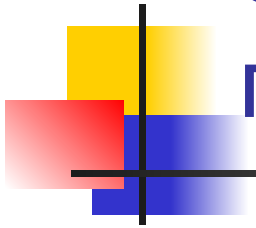
- Анти-паттерны (anti-patterns), также известные как ловушки (pitfalls) — это классы наиболее часто внедряемых плохих решений проблем.
- Они изучаются, как категория, в случае когда их хотят избежать в будущем, и некоторые отдельные случаи их могут быть распознаны при изучении неработающих систем.

Анти-паттерны в объектно-ориентированном программировании



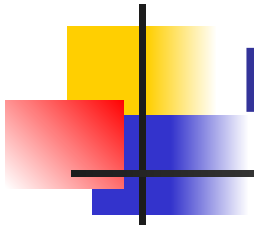
- Базовый класс-утилита (BaseBean): Наследование функциональности из класса-утилиты вместо делегирования к нему
- Вызов предка (CallSuper): Для реализации прикладной функциональности методу класса-потомка требуется в обязательном порядке вызывать те же методы класса-предка.
- Ошибка пустого подкласса (Empty subclass failure): Создание класса, который не проходит «проверку пустоты подкласса» («Empty Subclass Test») из-за различного поведения по сравнению с классом, который наследуется от него без изменений

Анти-паттерны в объектно-ориентированном программировании



- Божественный объект (God object): Концентрация слишком большого количества функций в одиночной части дизайна (классе)
- Объектная клоака (Object cesspool): Переиспользование объектов, чьё состояние не удовлетворяет (возможно неявному) контракту переиспользования.
- Полтергейст (компьютер) (Poltergeist): Объекты, чьё единственное предназначение — передавать информацию другим объектам
- Проблема йо-йо (Yo-yo problem): Структура (например: наследования) которая тяжело понятна вследствие избыточной фрагментации
- Синглетонизм (Singletonitis): Избыточное использование паттерна синглетон

Анти-паттерны в программировании

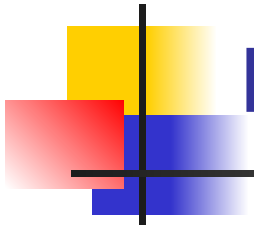


- Ненужная сложность (Accidental complexity): Внесение ненужной сложности в решение
- Действие на расстоянии (Action at a distance): Неожиданное взаимодействие между широко разделёнными частями системы
- Накопить и запустить (Accumulate and fire): Установка параметров подпрограмм в наборе глобальных переменных
- Слепая вера (Blind faith): Недостаточная проверка (a) корректности исправления ошибки или (b) результата работы подпрограммы

Анти-паттерны в программировании

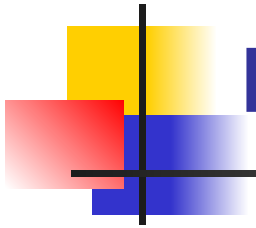
- Лодочный якорь (Boat anchor): Сохранение более не используемой части системы
- Активное ожидание (Busy spin): Потребление ресурсов ЦПУ (процессорного времени) во время ожидания события, обычно при помощи постоянно повторяемой проверки, вместо того, чтобы использовать систему сообщений
- Кэшированный сбой (Caching failure): Забывать сбросить флаг ошибки после её обработки
- Проверка типа вместо интерфейса (Checking type instead of membership, Checking type instead of interface): Проверка того, что объект имеет специфический тип в то время, когда требуется только определённый интерфейс
- Инерция кода (Code momentum): Сверхограничение части системы путём постоянного подразумевания её поведения в других частях системы

Анти-паттерны в программировании



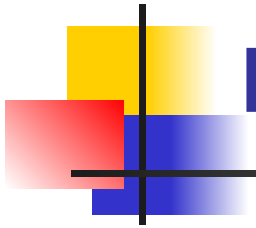
- Кодирование путём исключения (Coding by exception): Добавление нового кода для поддержки каждого специального распознанного случая
- Таинственный код (Cryptic code): Использование аббревиатур вместо полных (самоописывающих) имён
- Блокировка с двойной проверкой (Double-checked locking): Проверка перед блокировкой может выйти из строя в случае использования современного аппаратного обеспечения или компиляторов
- Жёсткое кодирование (Hard code): Внедрение предположений об окружении системы в слишком большом количестве точек её реализации

Анти-паттерны в программировании

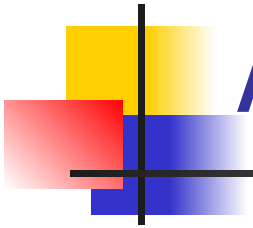


- Магические числа (Magic numbers): Включение чисел в алгоритмы без объяснений
- Процедурный код (Procedural code): Когда другая парадигма является более подходящей
- Спагетти-код (Spaghetti code): Системы, чья структура редко понятна, особенно потому что структура кода используется неправильно
- Мыльный пузырь (Soap bubble): Класс, инициализированый мусором, максимально долго притворяется, что содержит какие-то данные.

Методологические анти-паттерны



- Программирование методом копирования-вставки (Copy and paste programming): Копирование (и лёгкая модификация) существующего кода вместо создания общих решений
- Дефакторинг (De-Factoring): Процесс уничтожения функциональности и замены её документацией
- Золотой молот (Golden hammer): Сильная уверенность в том, что любимое решение универсально применимо
- Фактор невероятности (Improbability factor): Предположение о невозможности того, что сработает известная ошибка
- Преждевременная оптимизация (Premature optimization): Оптимизация на основе недостаточной информации
- Изобретение колеса (Reinventing the wheel): Ошибка адаптации существующего решения
- Изобретение квадратного колеса (Reinventing the square wheel): Создание плохого решения, когда существует хорошее



Анти-паттерны. Итоги

- Анти-паттерны (anti-patterns), также известные как ловушки (pitfalls) — это классы наиболее часто внедряемых плохих решений проблем.
- Они изучаются, как категория, в случае когда их хотят избежать в будущем, и некоторые отдельные случаи их могут быть распознаны при изучении неработающих систем.