Designmängel

übernommen bis auf Seiteneffekte, nicht mnemonische Benennung undunangemessener Gültigkeitsbereich, Spaghettilogik & verschachtelte Bedingung, Direktwerte

i. Ausgangspunkte

1. doppelter Code (Duplicate Code)
2. überlange Methode (Long Method)
3. ausladende Klasse (Large Class)
4. unübersichtliche Parameterliste (Long Parameter List)
5. mehrgleisige Änderung (Divergent Change)
6. verstreute Änderung (Shotgun Surgery)

ii Strukturierung

1. befreundete Datenelemente (Data Clumps)
2. fehlende oder objektlose Feinstruktur (Primitive Obsession)
3. Datensatzklasse (Data Class)
4. unnütze Klasse (Lazy Class)
5. Spaghettilogik

iii Vererbung

1. Fremdkopplung (Feature Envy)
2. wiederkehrende Case-Anweisung (Switch Statement) //auch ProcVariablen
3. parallele Vererbungshirarchien (Parallel Inheritance Hierarchies)
4. ungenutztes Erbe (Refused Bequest)
5. analoge Subklassen (Gegenmittel: Tease apart Inheritence)
6. ungenutzte Möglichkeit (Gegenmittel: Extract Hirarchy)

iv Koppelung und Schnittstellen

1. unerwünschte Seiteneffekte
2. umständliche Nachrichtenkette (Message Chains)
3. rigide Delegation (Middle Man)
4. stark gekoppelte Klassen (Inappropriate Intimacy)
5. Teilfunktionalität aus Bibliothek übernehmen (Incomplete Library Class)

v Kontext und Metainformationen

1. zu großer oder zu kleiner Gültigkeitsbereich
2. temporär genutzte Instanzvariablen (Temporary Field)
3. nicht genutzte Flexibilität (Speculative Generality)
4. übereifrige Kommentierung (Comments) & Direktwerte
5. nicht mnemonische oder inkonsistente Namensgebung (misnomer, Alternative Classes with Different Interfaces)

Umgestaltungsmöglichkeiten

I Grundlegendes

1. Routine extrahieren oder einsetzen
2. Temporärvariable ersetzen (parallele Zuweisung nur Lisp)
3. Zuweisung an Parameter entfernen
4. Umbenennen, Temporärvariablen separieren, unbenutzte Variablen entfernen (Kosmetik)
5. Rekursion versus Iteration
6. Algorithmus substituieren
7. Ablauflogik vereinfachen
8. Nil/Guard-Objekt einführen

II Kontextwechsel

1. Zugriffsbereiche schaffen/auflösen (With-Anwsg)
2. Gültigkeitsbereich erweitern,einschränken&wechseln
3. Instanzvariablen und Methoden übersiedeln, Klassen extrahieren oder einsetzen
4. langes Unterprogramm einkapseln (Replace Method with Method Object)
5. Klassen und Objektkontext
6. Aggregieren/Einebnen (geschachtelte Klassen)

III Datenorganisation

1. statische versus dynamische Datenobjekte (Change Value/Reference)
2. von Feldern, Listen, Vektoren und Mengen
3. einfache und doppelte Verkettung
4. Datensynchronisation und Konsistenz
5. von Aufzählungen, Objekttypen und Klassenreferenzen(virte Konstrktren, vol.Typ.)
6. eigenverantwortliche Allokation (Factory Method)

IV Parameterübergabe

1. Parameter hinzufügen/entfernen
2. Übergabeart festlegen (val, var, in, out, ..)
3. Seiteneffekte eliminieren
4. Funktionalitäten trennen (Farbe setzen und abfragen: für FN kontraintuitiv)
5. Methoden trennen/parameterisieren (Prozedurvariablen)
6. Objektparameter (Preserve Whole Object, Introduce Parameter Object)

V Kapselung

1. Eigenschaften (Introduce/Remove Setting Method)
2. Sichtbarkeit ändern (priv., publ, prot, :private classxy)
3. Schnittstelle einrichten (Hide Method, private&public, Extract Interface)
4. Superklasse aufräumen
5. spezialisierende Typumwandlung kapseln (Encapsulate Downcast)
6. bestehende Klasse konsistent erweitern

VI Sprachelemente und Entwurfsmuster

1. Nachrichtenketten, Delegation und Vermittler
2. Delegation und Vererbung
3. Ausnahmen, Flags, explizites Testen
4. volatile Typisierung (Typänderung zur Laufzeit)
5. andere Entwurfsmuster (Design Patterns: MVC, Iterator, …)