Ch7

* Requirements Modeling for WebApps
  1. Content Analysis（內容分析）
     1. 完整的內容範圍應由WebApp提供，包含文本、圖形和圖像、視頻、音頻資料
     2. 資料模組可以用於識別和描述每個資料物件
  2. Interaction Analysis（交互分析）
     1. 詳細描述使用者與WebApp交互作用的方式
     2. 可以開發使用者案例來提供這種交互作用的詳細描述
  3. Functional Analysis（功能分析）
     1. 作為交互作用分析的一部分創建的使用方案，定義了應用於WebApp內容的操作，並暗示其他處理功能
     2. 詳細描述所有操作和功能
  4. Configuration Analysis（配置分析）
     1. 詳細描述WebApp所在的環境和基礎架構
* The Interaction Model
  1. 4個組成要素
     1. use-cases
     2. sequence diagrams（時序圖）
     3. state diagrams （狀態圖）
     4. a user interface prototype（使用者界面原型）
  2. 都是重要的UML符號
* Data Flow Diagramming: Guidelines
  1. 所有圖標都有一種有意義的名字
  2. DFD是經由許多層級的細節演變而成的
  3. 總是從上下文層級圖（0級）開始
  4. 總是顯示0級的外部實體
  5. 在箭頭上標示資料
  6. 不呈現程序邏輯
* Behavioral Modeling
  1. 列出系統的不同狀態
  2. 指出系統如何轉換
     1. 指出事件
     2. 指出行動
  3. 畫state diagram或sequence diagram
* Patterns for Requirements Modeling
  1. Software patterns是一種獲得領域知識的機制，允許在遇到新問題時重新應用
     1. 領域知識可以應用於相同領域的新問題
     2. 由pattern所捕獲的領域知識可以類比應用於不同的領域
  2. 不必創造，只需發現
  3. 一旦模式被發現，就記錄下來

Ch8

* Design

1. 好的軟體設計應該存在：
   1. Firmness（堅固）：一個程式中不應存在任何會抑制功能的Bug
   2. Commodity（商品）：一個程式應符合它的預期目的
   3. Delight（愉悅）：使用程式的經驗應令人愉快

* OO Design Concepts

1. Design classes
   1. Entity（實體）classes
   2. Boundary（界線）classes
   3. Controller（控制）classes
2. Inheritance：所有superclass的責任都會被subclasses繼承
3. Messages：刺激接收物件的行為發生
4. Polymorphism：是一個能大量減少擴大設計所需工作量的特性

* Design Principles
  1. 設計過程中不該受限於眼界
  2. 設計應該可追溯到分析模型
  3. 設計不應該重新發明輪子
  4. 現實世界中，設計應量減少軟體和問題間的知識距離
  5. 設計應顯示均勻性和集成性
  6. 設計應結構化以適應變化
  7. 即使遇到異常數據，事件或操作，設計也應結構化以輕度降級
  8. 設計不是coding，coding不是設計
  9. 設計應在創建時評估質量，而非之後
  10. 設計應被審查，以最小化概念（語義）錯誤