

模擬螞蟻

——以物件導向分析遺傳演算法——

前言	1
BOOCH 的方法論 (BOOCH'S METHODOLOGY)	2
需求分析 (REQUIREMENT ANALYSIS)	2
系統功能敘述 (SYSTEM FUNCTION STATEMENT)	2
遺傳演算法 (GENETIC ALGORITHM)	2
領域分析 (DOMAIN ANALYSIS)	2
找出主要的類別 (IDENTIFYING MAJOR CLASSES)	2
找出關係 (IDENTIFYING THE RELATIONSHIPS)	2
找出物件行為 (IDENTIFYING THE OPERATIONS)	2
劇本 (Scenario)	2
加入行為 (Classes with Operations)	2
屬性與繼承 (ATTRIBUTES AND INHERITANCE)	2
驗證及反覆修改 (VALIDATION AND ITERATION)	2
系統設計 (SYSTEM DESIGN)	2
定義初始架構.....	2
計劃可執行版本.....	2
發展可執行版本.....	2
檢討與策進.....	2

前言

- 就讀銘傳資管系時，選修了何祖鳳老師的“軟體工程”中，我們這組決定以“模擬螞蟻”這個題目作為期末報告。
- 當初是以傳統結構化方式的軟體工程來分析、設計整個系統，產生一些描述性的、大方向上的文件：包括使用者介面，及中、上層的控制流程等。
- 就讀淡江資工系時，在洪文斌老師的專題課中，被要求在網路上找一個 JAVA Applet 的程式，研究後寫一篇報告上來。
- 於是在網路上找到 Mark Miller 所撰寫的“Manna Mouse” JAVA Applet，在看了其原始碼後，覺得其程式的架構實在是太雜亂了。
- 於是就決定自己利用物件導向分析與設計的方法來將整個程式重新設計過。
- 如此，一方面可以加深自己對物件導向分析與設計方法的熟練程度；另一方面我對於這個 Applet 所使用的遺傳演算法也有很大的興趣，想藉此有更深的探究...

Booch 的方法論 (Booch's Methodology)

- Grady Booch 一向在物件導向 (Object Oriented, OO) 的方法論上執領導的地位，他在 1986 年率先提出 OO 開發方法，開啟物件導向分析與設計的研究。
- Booch 方法在進行分析時，不只是針對問題本身作分析，也針對問題的領域進行分析。因為在同一領域的問題，有很多非常類似。實在不需要對這些類似的問題，每次都一一地重新作類似的分析。
- 對於問題的整個領域作一徹底的分析，雖然在一開始的時候需要耗費相當的心力，可是隨著分析的進行，我們對於整個領域就會有更深入的瞭解，最後不但累積了領域相關的知識，也完成了富彈性、高度可重複使用的軟體元件，對於將來開發新的系統將帶來非常大的助益。
- Booch 的開發方法非常強調領域內的可重用性 (Reuseability)，其將軟體開發分成以下三個階段：
 1. 需求分析 (Requirement Analysis)
 2. 領域分析 (Domain Analysis)
 3. 系統設計 (System Design)
- 上面三個步驟並不是像傳統瀑布式的分析設計方法般僵硬地要求一定要一步步地執行；而是允許在步驟間回溯 (backtrack)，進行反覆地修改與調整，直到系統符合需求為止。

需求分析 (Requirement Analysis)

系統功能敘述 (System function statement)

- 在看了 Mark Miller 的 Manna Mouse Applet 的執行後，覺得畫面上的黑點與其稱之為老鼠，不如將之當作螞蟻還更像些。
- 系統執行時，使用者可以在模擬的蟻群間丟入一些食物，螞蟻發現食物後，會有聚集到食物上的傾向；漸漸地，螞蟻們在把食物吃光後，就又自顧自地散開、離去。
- 所謂的螞蟻，不過是畫面上的黑點。每隻螞蟻都有自己的“基因” (gene)，基因決定了螞蟻現在所在的座標位置。
- 螞蟻在吃了食物後，其適應度 (fitness) 就會增加。
- 螞蟻的移動，其實是利用“遺傳演算法” (Genetic Algorithm, GA) 的世代 (generation) 交替來完成。
- 在每一個世代交替時，系統會選出一些適應度較高的螞蟻來繁殖下一代。下一代螞蟻的位置，是決定於其父母座標基因的互換 (crossover) 後，再經過突變 (mutation) 完成的。

遺傳演算法 (Genetic Algorithm)

- 根據生物學的说法，曾經在地球上出現過或現存的所有生物都是由構造簡單的、原始的生物，經由長期的演化而逐漸產生的。
- 生物演化的理論，在早期可分成：相信後天獲得的性狀可遺傳給後代的拉馬克 (Lamarck) 學說，及相信後天性狀不可遺傳給後代的達爾文 (Darwin) 學說兩大派。
- 由於後來新證據不斷地出現，及這幾十年來分子生物學的快速發展，生物學上已經知道自然界生物的演化機制是採用達爾文式的。雖然有學者提出人類文化的演進可以看成是採用拉馬克式的演進方式，不過這不是這裡所要討論的重點。
- 達爾文的生物演化論可分成以下幾個步驟：
 1. 遺傳變異 (genetic diversify)：個體間因遺傳基因型 (genotype) 的差異而顯現出各式多樣的表現型態 (phenotype)。
 2. 自然選擇 (natural selection)：自然界對於不能夠於所在環境中適應良好的個體發生淘汰的現象。
 3. 繁殖複製 (reproduction)：未被淘汰的個體經由繁殖，產生許許多多的後代。這其實是一種選擇放大 (selection amplify) 的情形。

新產生的後代間又稍有不同而產生了遺傳變異，因此不斷地變異—選擇—複製，變異—選擇—複製，變異—選擇—複製.....一直反覆循環下去即發生了生物的演化。

- 遺傳演算法在電腦科學上屬於演化式計算 (Evolutionary Computation) 的一環。遺傳演算法其實是以電腦來模擬生物演化的過程。通常用來解一些對問題解法沒有很充分瞭解的例子，或在用傳統的解法成本過高的時候使用。

領域分析 (Domain Analysis)

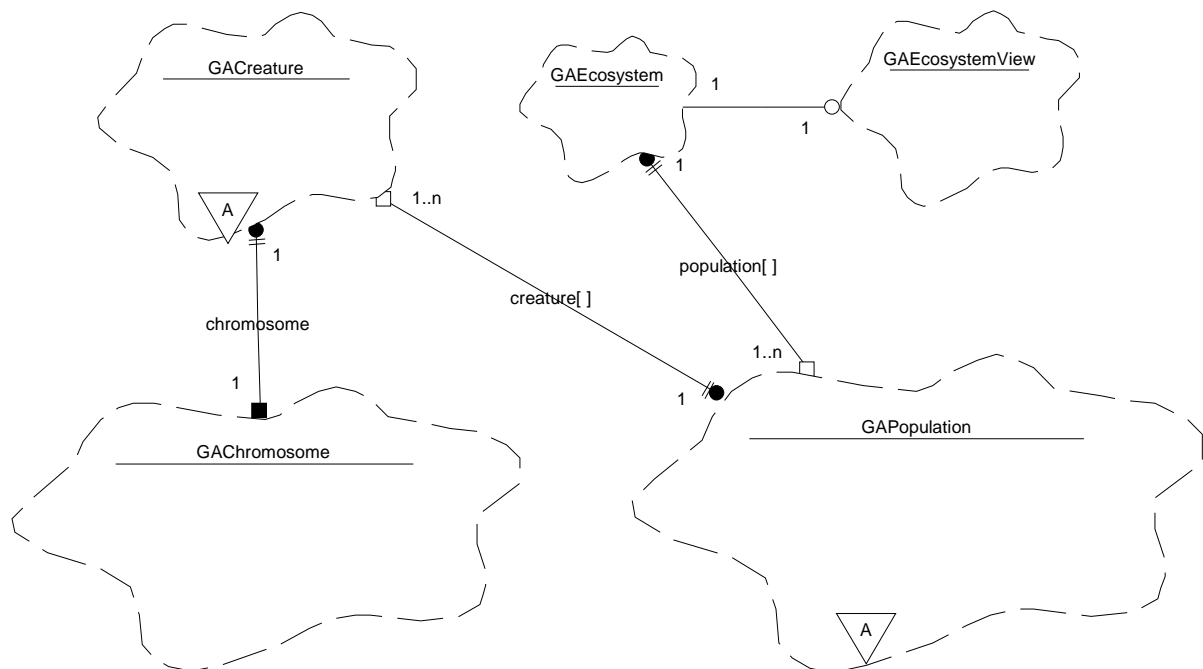
找出主要的類別 (Identifying Major Classes)

➤ 經過初步的分析，找到了以下幾個類別：

- GAEcosystem：這個類別被對應到生態系統 (ecosystem) 上。
- GAPopulation：這個類別對應到生態系中的一個族群 (population)。
- GACreature：每個族群有很多同種的生物 (creature) 個體。
- GACHromosome：每個生物個體都有自己個別的染色體 (chromosome)。
- GAEcosystemView：這是個輔助類別，用來在螢幕上顯示整個生態系的模擬過程。

找出關係 (Identifying the Relationships)

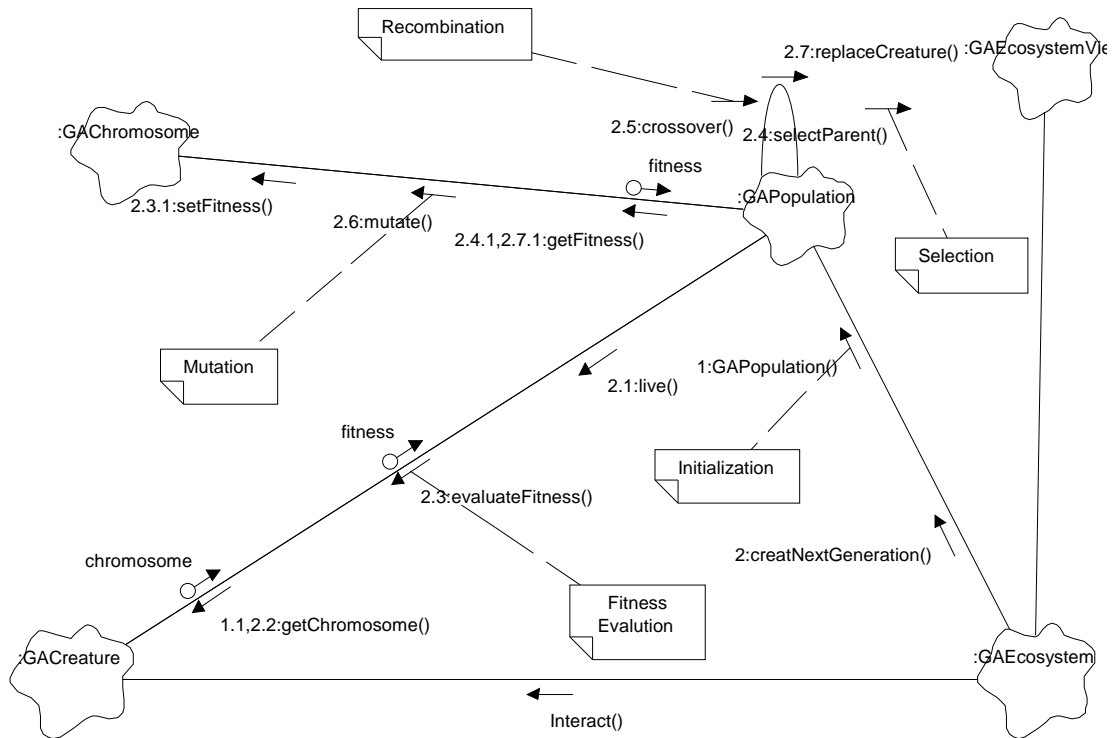
➤ 在發現了系統所需要的類別後，接下來當然要定義類別間的關係，其定義如下圖：



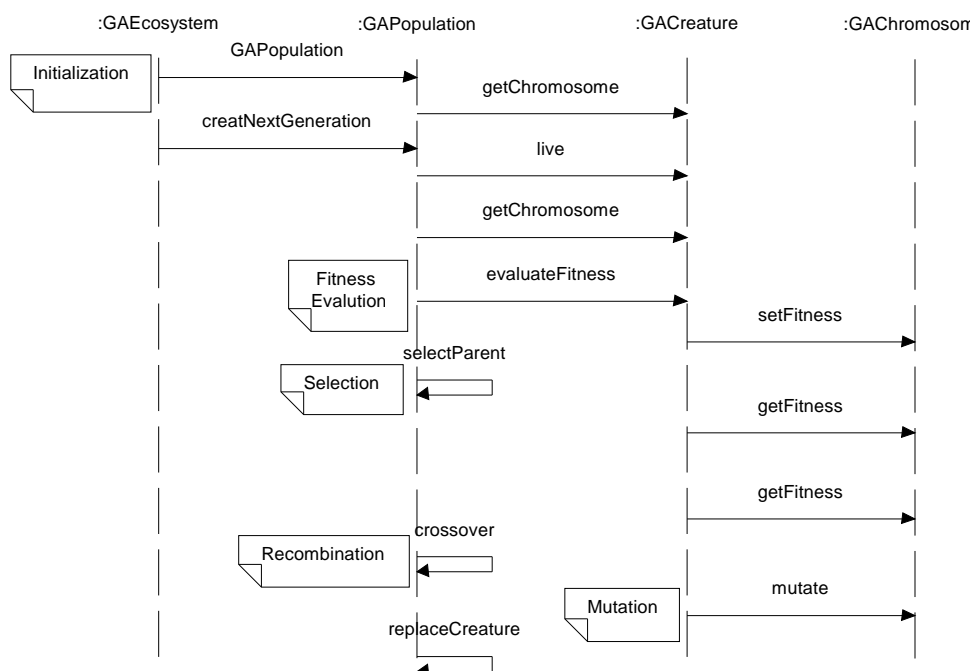
找出物件行為 (Identifying the Operations)

劇本 (Scenario)

- 要找出物件的行為，我們可以利用物件圖將物件間可能的行為反覆地“排演”，就好像在編排劇本一般：

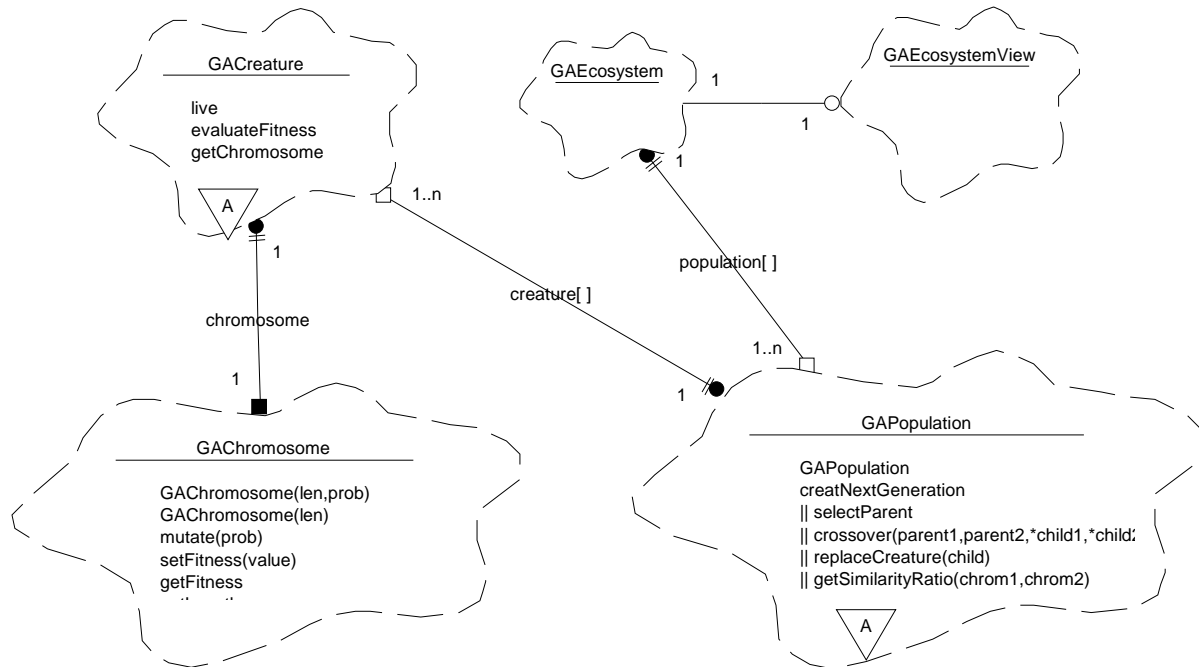


- 為了更精確地表達物件間的互動，繪製了物件互動圖如下：



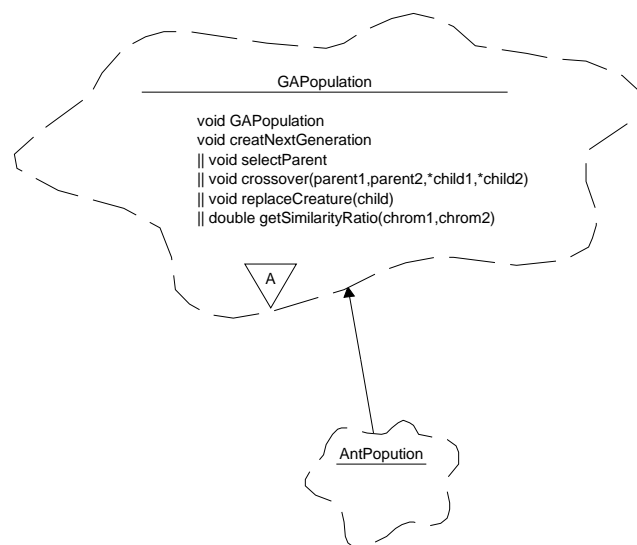
加入行為 (Classes with Operations)

➤ 在編排好物件的劇本後，我們就可將物件所屬的類別加上一些行為的定義：



屬性與繼承 (Attributes and Inheritance)

➤ 接下來我們可以定義生物類別與螞蟻類別間、族群類別與蟻群類別間的繼承關係：



驗證及反覆修改 (Validation and Iteration)

- 領域分析要經過不段地驗證、反覆地修改，不停地進行推敲，以期高度表現出領域的相關知識，建構高度的可重用性。

系統設計 (System Design)

定義初始架構

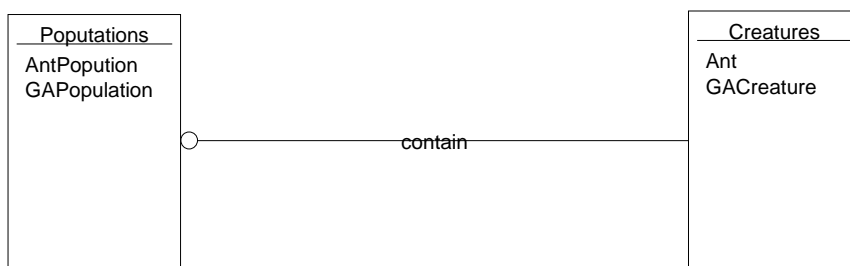
- 通常，強調未來的擴充及維護性，常使整個工作完成的日期延誤；可是，輕率地定義架構，卻使未來在維護及擴充時發生困難。
- 不過就學習的角度來衡量，對問題作徹底分析，當然可以習得相關領域的知識，況且分析的過程都有文件留下來，日後不管是自己或別人都可以更快的認識這個問題領域。

計劃可執行版本

- 發展可執行版本的目的，是要驗證關鍵部分的分析是否能正確無誤地執行，並找出當初領域分析時，考慮不夠周詳、或是不正確的地方。
- 而且也可以看看這個軟體是否合乎需要。

發展可執行版本

- 實際建構軟體時可將類別進一步分類，分別放到不同目錄 (category) 或封包 (package) 中：



- 我們這裡當然是要用 Java 來完成整個蟻群模擬系統，雖然由前面系統功能敘述中看來，這個系統根本和實際的蟻群完全不同，不過這裡將其發展出來的目的只是為了驗證領域分析階段所設計的類別是否正確無誤，並可以應用於實際的問題中。

檢討與策進

- 這次我使用的是 Object Domain Systems 公司的 Object Domain Version 1.19，這套 CASE Tool 來繪製 Booch 方法論的 Notation 圖。
- 這套 CASE 雖然也支援 OMT 的方法，不過，OMT 的方法有傳統結構化分析設計的影子，所以我比較偏好 Booch 的方式。
- 事隔兩年，物件導向分析設計的軟體發展方式已逐漸普遍，Booch 方法論也結合了 OMT 及 Use Case 而形成的 UML 的標準。UML 的規格書也已出爐、大致底定，成為業界的標準。
- 遺傳演算法要結合其他應用才能發揮功能，其可以用在很多地方，如用來尋找數學函式的極值問題、調整類神經網路 (ANN) 的神經元間連結點權值、設定模糊控制 (Fuzzy control) 上的歸屬函數之參數，或分類者系統 (Classifier system) 中的分類元規則之可信度等。
- 所以只要定義好介面，這個軟體元件必定可使用在許多不同的系統上。
- 不過當初在分析設計時，一方面是時間的因素，一方面也是對於問題沒有作適當的取捨，所以很多設計都還嫌不夠完善，還有很多改善的空間。