國立交通大學

資訊管理研究所

碩士論文

軟體元件保護方法之研究

A Study on the Protection Approach of Software Components

研究生：余新平

指導教授：羅濟群　教授

中華民國九十四年六月

軟 體 元 件 保 護 方 法 之 研 究

A Study on the Protection Approach of Software Components

研 究 生：余新平　　Student：Hsin-Ping Yu

指導教授：羅濟群　　Advisor：Chi-Chun Lo

國 立 交 通 大 學

資 訊 管 理 研 究 所

碩 士 論 文

A Thesis
Submitted to Institute of Information Management
College of Management
National Chiao Tung University
In Partial Fulfillment of the Requirements
For the Degree of
Master of Business Administration
in
Information Management
June 2005
Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

軟 體 元 件 保 護 方 法 之 研 究

學生：余新平　　指導教授：羅濟群 教授

國立交通大學資訊管理研究所 碩士班

摘　要

軟體的使用權控管對軟體產業及組織是一個重要的議題，未經授權的使用會造成軟體公司的投資無法回收，對組織則可能洩漏原先已嚴密保護的軟體功能或資料。過去為軟體保護所發展的研究與技術，較不易破解的方法多依賴網路伺服器做線上的使用權檢查，可能造成正常的軟體因為不正常的網路而無法使用。加上近年來軟體系統朝元件化發展，過去針對整個軟體系統考量而設計的研究及技術便無法真正達到使用權控管的目的。

本文將發展上述問題的解決方法視為軟體系統開發的專案，因此遵循RUP(Rational Unified Process)的流程，逐步探討元件保護方法的需求、解決方案。所提出的方法以非對稱加密法限制元件僅能在單一已申請的電腦上使用，以智慧卡保護私密金鑰不被複製，即使元件被不當複製到其他電腦上亦無法使用；在檢查使用權時不需要依賴網路，建立一套元件自我檢查的機制，讓使用者或應用程式在使用此類受保護元件時，不需要改變既有習慣。這個方法若是回過頭應用在整個軟體系統上，亦能保護軟體系統的使用權。

關鍵字：元件保護、軟體保護、軟體盜版、RUP

A Study on the Protection Approach of Software Components

Student：Hsin-Ping Yu　　Advisors：Dr. Chi-Chun Lo

Institute of Information Management
National Chiao Tung University

ABSTRACT

The protection of software usage privileges is an important issue to software market and organizations. Illegal usage of software will cause the investment of a software company to be nothing, and cause a organization to reveal the secret software functions. There are a few of research papers and techniques to solve these problems, but some of them which are not easy to crack depend on a server to check the usage privileges. If the network crashes, the software or components can't work any more. In recent years, a system is decomposed to many components, and then the existing approaches that target on whole system can't indeed provide the control of usage privilege.

This thesis treats the developing of above problems as a software project. So we follow the RUP processes to discuss the requirements and the solution of component protection. This approach limits the components only running on registered computers by adopting the asymmetric encryption algorithm and protects the private key from copying by storing it in a smart card. The designed checking process of usage privilege will not depend on network, and users or applications need not to be aware of the extra protection of components by developing the self-checking mechanism.

Keywords: component protection, software protection, software piracy, RUP

目錄

表目錄

圖目錄

一、緒論

1.1 研究背景與動機

目前軟體系統的開發趨向元件化的方式，軟體市場不再是僅能以一完整的系統出售，一組功能明確且能達成一特定目標的元件亦能成為銷售的產品。在組織內的系統也漸漸解體成一個個小元件，然後再組裝成特定的系統，以使同一元件在不同系統間重用達到最大效用。

軟體為了保障商業利益或是在組織內限定使用對象，都必須具備使用控管的能力。但是綜觀目前軟體的保護方式，有的太過簡略易於破解，有的操作上太複雜且使用不易，或是必須仰賴網路上的伺服器做使用權線上檢查。而且對於傳統單一系統與目前廣為使用的元件，其保護方式更需要進一步修改以適用於元件環境，例如使用控管的程序必須標準化以簡化開發、授權證明的存取必須簡易才能讓許多元件在同時間運作良好(如光碟母片檢查方式，若是在不同系統使用時必須抽換不同的光碟片)。

在過去可能一年才開發一個或二個完整的系統，因此可以在每一個系統中獨立實作使用控管的程序，甚至將此程序寫死在系統中都沒有關係，畢竟完整的系統並不會經常升級，而且系統具有單一入口，即使更新系統的某些部分，這些驗證的程式碼亦不需變更。相較於元件而言，同一期間所開發的元件數量將遠超過傳統的系統，並且具有獨立的入口，如果每個元件都要重新實作這些驗證的程序，將會提高開發成本。像目前影音文件以 DRM 方式管理授權，並不需要大幅改變我們製作這些影音文件的方式，而且對於使用者而言，只需簡單的驗證程序就可以使用，因此本文探討的就是如何讓元件能夠像這些影音文件輕易地具備使用控管能力，並且在開發與使用時都不需大幅改變原有習慣，至少希望盡量使必要的改變幅度最小化。

1.2 研究目的與預期貢獻

由於目前的軟體使用控管保護方式係針對傳統完整系統設計，因此本文首先會探討這些設計方式的優缺點。再者元件已與傳統系統有一些本質上的差異，因此在設計方式的選擇上必須有一些不同的考量。在比較過以上的方式之後，本文將提出一套保護方

法，並討論其中的解決方案之優缺點，以應用在元件的授權保護上，並期望同樣的方法用在傳統完整系統的保護上亦能奏效。

1.3 研究適用範圍與限制

本文探討的動機是來自於元件的使用控管保護，希望最後所設計出來的架構能夠應用在元件的商業市場上，不過為了更能集中本研究的焦點，因此討論解決方案時以公司內的元件使用管理為主要考量。例如本研究的設計中，選擇以智慧卡儲存私密金鑰。雖然說智慧卡的使用已漸漸擴展，但仍然是在起步增長的階段，而且未來更會走向一卡多用的時代，因此要讓本研究的架構普及化，在目前會顯得成本較高。但如果是在單一組織或公司內，智慧卡的製作與管理都較為容易達成，而且目前來說，在組織內元件使用控管的需求更高於將元件販售於大眾市場中，因此為了更容易說明與設計，本文選擇以在公司內使用來做為實作對象。

另外目前各種的軟體保護其實都有破解的方法，有些因為其保護機制的設計本身就以容易實作為考量，而犧牲了保護的嚴密程度；而大部分設計的非常嚴密的保護，也都可能因為駭客行為而遭到破壞。此處所指的駭客行為主要是指中斷程式碼的方式，駭客在程中執行中竄改已載入記憶體中的資料，以將檢查授權的程式碼略過，或回傳已檢查成功的假訊息。此類的行為在本研究中尚無法解決，不過這類防範工作若由作業系統層級(例如 Microsoft Windows XP Service Pack 2 的資料執行防止)的記憶體防護的技術完成，比起從一般軟體程序上防範，其效率與效能上都會更好。因此本文假設略過程式碼檢查以及竄改記憶體中的資料是非常困難的。

1.4 章節規劃

本文第二章開始先介紹目前對於軟體保護的研究，討論這些方法的優缺點及應用在元件上是否適用，再介紹兩項會應用在本文架構上的主要技術：公開金鑰加密方法與數位簽章、智慧卡的應用，最後介紹 UML 與 RUP 做為本研究在設計保護方法時的遵循流程。第三章以一個虛擬的軟體公司為例，開發適用於此公司的元件保護功能，由於遵行的是 RUP 的系統發展方法，因此本章將說明 RUP 如何具體地指導開發出元件保護方法。第四章依第三章所述的步驟，逐步分析元件保護方法的需求、設計系統架構、實作系統雛型，以完成對元件保護方法的研究。第五章是本文方法貢獻的整理以及未來可以

繼續進行的工作。

二、文獻探討

2.1 以元件為保護對象

軟體元件的想法是來自於工業界的硬體設計及組裝方式，就像電腦主機每一部分的零件[1]一般，為的就是便於開發、除錯、更新等。使用元件的優點是顯而易見的，且在工業界已行之有年。而軟體元件的概念雖然簡單，不過不同的研究對於元件有不同的定義，N. S. Gill[2]在他的一篇研究報告中整理目前廣泛使用的定義後，提出他自己的定義：

A software component is a coherent package of software implementation that:

1. *Carries out a set of related services or functions*
2. *Offers well-defined and published interfaces.*
3. *Offers services that are accessible through its interfaces only.*
4. *Is reusable.*
5. *Can be independently developed and delivered.*

對於第 5 項的定義與本研究所謂的元件最相關，這一項定義清楚地說明元件是一個獨立發展的軟體單元，而且可以單獨被散佈。目前以元件為基礎的系統開發方式已蔚為主流，在 Kruchten《The Rational Unified Process, An Introduction》[3]一書中揭示了元件化開發方式的重要性及優點：

> 設計一個有彈性的架構很重要，因爲它可以很經濟地提昇再使用性（Reuse）、清楚劃分開發團隊工作、把容易改變的硬體與軟體相依性（Dependency）獨立出來，改善可維護性（Maintainability）。
>
> 對軟體架構（Software Architecture）來說，以元件爲基礎（Component－based）的開發方式是一種很重要的開發方法，它讓各種商業元件

[1] 元件可以再包含其他元件，例如主機板是一個主要的元件，它是由其他更小的元件(如 BIOS、電池)組成。而主機板這樣的元件必定會提供一些介面以能便利地於其他元件組合成個人電腦，如 AGP 的介面與顯示卡溝通、IDE 介面與硬碟機及光碟溝通等等。

[2] N. S. Gill, P. S. Grover, “Component-based measurement : few useful guidelines”, November 2003, ACM SIGSOFT Software Engineering Notes, Volume 28 Issue 6.

[3] Philippe Kruchten, 《The Rational Unified Process, An Introduction》, 2nd Ed., Addison Wesley, 2000, p.10。本段引文出自該書中文版(趙光正譯)。

（Component）得以再使用（Reuse）或客製化（Customization）。……元件擴大再使用的規模，使得系統可以由現有零件、協力廠商提供的現成（Off-the-Shelf）零件、解決特定領域問題的零件或整合其它零件的零件組成。

也因此組織中的資訊系統充斥大大小小的元件，而每一個元件都可能在一或多個資訊系統中運作。

元件的一項重要優點就是能夠重覆使用，但試想以下的情境：組織內的會計系統有一項重要的功能是觀看某位員工的薪資，而這項功能只有特定的人員才能觀看，所以在系統啟動的時候，會要求使用的人員登入以驗證身分，以保護這項功能不被未經授權的人存取。在過去，系統是一個完整的黑盒子時，這個方法是可行的，未經授權的人員無法避過登入程序而使用觀看薪資的功能，但是現在由於此會計系統是以元件開發而成的，觀看薪資的功能被發展成一個獨立的元件，雖然系統啟動時仍提供身分登入的功能，但是未經授權的人員卻可以另行發展一個程式來存取此關鍵元件，因此原先的保護機制不復存在。如下圖所示。

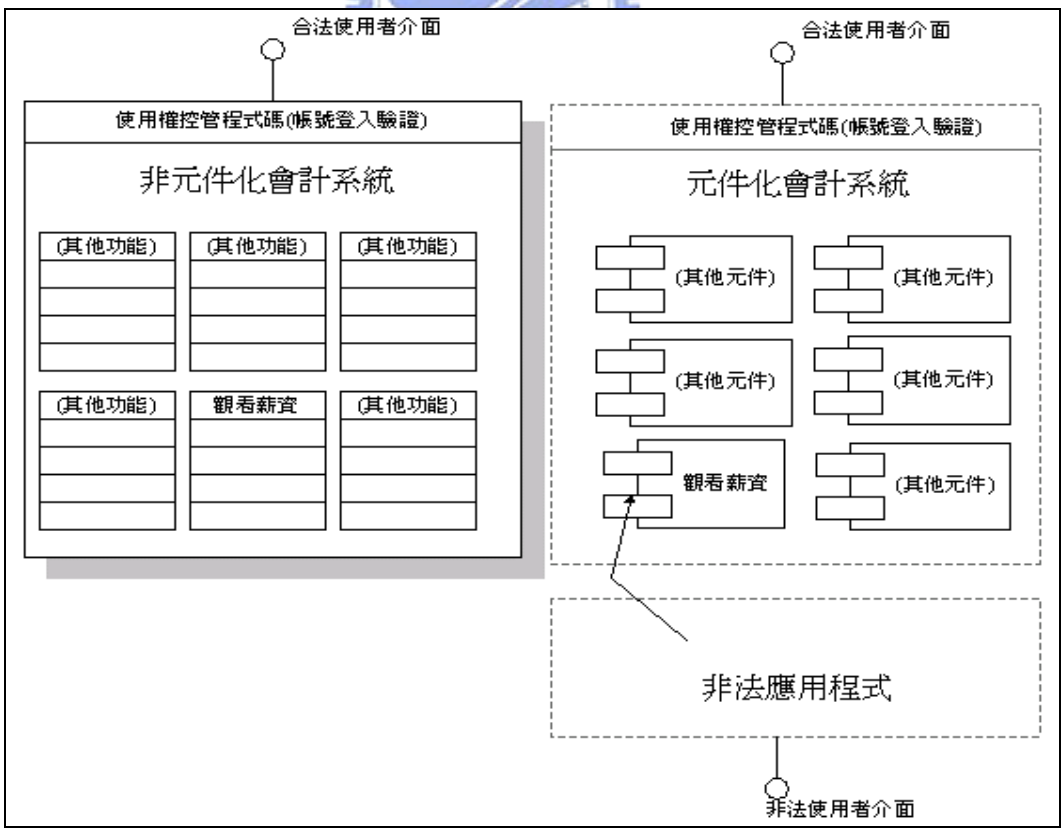

圖 1：非法使用需保護功能

上圖中的左邊是傳統非元件化的會計系統，提供一合法使用者介面以存取該系統，因此所有的系統功能都必須透過帳號登入驗證後才能使用；右邊則是元件化後的會計系統，其系統邊界其實是虛擬的，雖然看起來需要透過此合法使用者介面存取此系統功能，但是由於每一個元件都有對外公開的介面，因此非法使用者只要另行建立一個不具身分驗證功能的系統(圖右下，提供非法使用者介面)，再引用該觀看薪資功能元件即可非法使用該功能。

因此元件可重覆的這項優點，反而成了非法使用者得以輕易使用該功能的缺點。所以本研究的保護對象，將鎖定在元件上。

2.2 軟體保護相關研究介紹

目前的相關研究焦點都是以一般性的軟體為保護對象，以元件為對象的研究在本文研究期間尚未發現，不過元件亦是軟體的一種特殊形式，因此這些軟體保護的方法應用在元件開發上亦能發揮作用，但由於元件的特性既不全等於一般性軟體，在設計方案的選擇上就有一些不同的考量，這一點會在後文中提出比較。

本研究期間所探討的文獻中，有提出具體保護機制的有以下數篇：

1. 林清展，「軟體使用權控管機制之研究」，靜宜大學，碩士論文，2001。
2. 林祝興，李鎮宇，「網路軟體保護方法之研究：一次安裝方案」，二〇〇〇網際網路與分散式系統研討會論文集 I，438-441 頁，台南，2000。
3. Patrick C.K. Hung, Kamalakar Karlapalem, “Security and Privacy Aspects of SmartFlow Internet Payment System,” *Proceedings of the 32nd Hawaii International Conference on System Sciences,* 1999.
4. Shiuh-Pyng Shieh, Chern-Tang Lin, Shianyow Wu, “Optimal Assignment of Mobile Agents for Software Authorization and Protection”, *Computer Communication*, 22, pp. 46-55, 1999.
5. Tomas Sander, Christian F. Tschudin, “On Software Protection Via Function Hiding”, *2nd International Workshop on Information Hiding,* 1998.

林清展(1)的方法是軟體在使用前必須先註冊使用權，並將使用者資訊與使用權記錄

在伺服器資料庫中，使用時由受保護的軟體傳送使用權代碼到伺服器上以驗證使用權是否正確。此方法的優點是除了能控管使用者的身分，尚能控管其使用次數，但軟體執行時需要依賴網路以及另一台伺服器檢查使用權，當網路或伺服器發生故障時，該軟體將無法正常使用。

林祝興、李鎮宇(2)的方法則是將軟體放置於伺服器上，當使用者欲下載時以伺服器會取得用戶端時間產生一次安裝套件(已經過加密的套件)，待使用者執行安裝時再將時間戳記上傳取得套件解壓縮金鑰，然後得以解壓縮到暫存目錄安裝，安裝後即將暫存目錄刪除。但使用者如果在套件已解壓縮、安裝完成之間將暫存目錄的檔案先行複製留存，此保護機制即遭到破解。

Patrick(3)原是討論網路交易服務的安全性流程，但其機制可以應用在軟體的授權控管上。使用者於購買軟體後，必須向軟體廠商的伺服器取得安裝與使用的 License Key，使用者必須將此 License Key 放在智慧卡中，License Key 包含此智慧卡的唯一發行碼(issuer code)、網路 IP 位址、使用期限等資訊，以確保此 License Key 只能在單一卡片及單一機器中使用。額外的資料則可以控管軟體的使用範圍，例如使用期限。廠商亦可在智慧卡中放入一個計數器(counter)，用以控管消費者使用軟體或安裝的次數。但此方法每一個軟體都必須有一個 License Key， 而目前智慧卡的空間有限，但元件數量卻會不斷增加，當元件數量的 License Key 容量大於智慧卡空間時，就必須再使用第二張智慧卡，而元件又常常同時執行，因此恐怕會造成需不斷抽換智慧卡的困擾。此方法亦未討論 License Key 的偽造問題，如果使用者能夠讀出智慧卡的內容，在修改過後再存回智慧卡，則驗證時使用的是遭竄改的 License Key，當然就無法達到正確驗證的目的。

Shiuh-Pyng Shieh 等(4)將程式切割成二部分，一部分在使用者端執行，一部分在代理伺服器(proxy server)上執行。每個程式的切割均需手動進行，必須慎選哪些是關鍵的功能放在代理伺服器上，哪些是一般性的功能可以放在使用者端執行，使用者端的程式必須知道哪些功能在代理伺服器上，並負責呼叫及接收的過程。

Tomas(5)透過加密的方式保護重要函數的內容，程式中的函數會將該函數的輸出結果加密，使用者取得密文後，需送回給程式原創者，經原創者解密後再傳回給使用者，這一段加解密的過程可以是自動化也可以是人工操作，而授權控管的方式是由原創者(或伺服器)來驗證傳送者的身分。以上兩個方式(4, 5)除了需依賴另一台伺服器之外，程

式的開發是一項重大的負擔。

以上的文獻摘要以及目前商業應用上的軟體保護方式整理於下表 1 中。

表 1：相關文獻摘要及其他軟體保護方式[4]

防安裝	在安裝時檢查授權	一旦安裝後(或在解壓後)就可能被複製。檢查程序寫死在程式內	序號、一次安裝方案(2)、安裝記錄磁片
防使用	在使用時檢查授權	無法控管使用授權，不適合大量不同程式	檢查 keypro 或母片(每一程式需有個別的 keypro 或母片)
		智慧卡空間有限，不適合放置大量的 Key	在智慧卡中放置 License Key(3)
		網路為必備條件。檢查程序寫死在程式內	先註冊使用權，使用時線上檢查授權(1)
防複製	保護放置程式媒體	無法控管使用授權	光碟或磁片防拷
輸出加密(5)	關鍵函數的輸出先加密，再透過原作者檢查授權後解密傳回，使用上不便		
切割程式(4)	分割成兩部分，一部分在使用者端執行，一部分在 proxy server 上執行。每個程式的切割均需手動進行，不適用在大量不同程式		

從以上的討論中，可以整理出目前軟體保護的缺點有以下幾點，這些都是本文所提出的架構希望解決的問題。

1. 無法達到授權控管(如防安裝(2)、防複製)，無法限制在特定電腦上使用。
2. 必須仰賴網路(1)。
3. 程式有部分或全部需由 server 執行(4)(5)。
4. 授權資訊存放方式不適用大量元件，新增及更新麻煩：
   (1). 硬體鎖 - keypro、母片。
   (2). 智慧卡 - 空間有限，不適合放大量的 license key(3)。
5. 控管的程序寫死在程式中，不適合大量開發。

[4] 本表格的分類方式取自參考文獻[3]。輸出加密與切割程式的方法亦可分類為防使用。

2.3 智慧卡

智慧卡具有獨立運算及儲存能力的微形電腦，通常以晶片的方式崁在如信用卡一般大小的卡片上，再透過讀卡機與智慧卡上的微形電腦溝通。

如果不考慮智慧卡提供的運算能力，單純就儲存資料這一點，就比傳統磁卡好。其一是容量，傳統磁卡的容量僅 140bytes，但智慧卡的容量卻可高達 64Kbytes。其二是智慧卡不容易被複製，因此只要卡片不遺失，其上的資訊就不會被盜用。

而且新型的智慧卡上可以有保護區域，使用者若未經授權，就沒有任何方式可以將這個區域的資訊讀出，因此就能保護一些重要但又不必與使用者互動的資訊。智慧卡內有獨立的運算能力與儲存空間，因此可以在上面開發應用程式。如果製卡者將重要的加密鑰匙放置於此保護區域，使用者僅能使用卡片提供的解密功能為訊息解密，卻無法將其中的鑰匙取出，就更能避免金鑰被取出而盜用。[5]

在本文的架構中，需要一個能夠安全地儲存私密金鑰的媒體，如果選擇以定製的硬體來做，則此架構就不容易擴張，而且由於使用量少，其成本必然提高，相較於智慧卡這種已漸趨普及的媒體，就失去競爭優勢。因此本文以智慧卡做為私密金鑰的儲存媒體，在未來，或許還能應用其強大的功能，使整個授權與驗證的架構更趨安全。

2.4 以非對稱式加密法驗證文件的發行者與限制閱讀對象

在本文提後續提出的架構中，元件授權書是用來驗證使用授權的唯一憑據，必須保證不被竄改或偽造，因此會使用到二次非對稱加密法來加密授權書，目的有二：一是確保只有已授權的使用者能夠解開此授權書，就能限制使用對象；二是確保授權書是由伺服器發出，否則非法使用者只要知道授權書內的格式，就能偽造授權書，授權書的憑證意義就失去了。因此以下略為介紹非對稱加密法的概念，以方便後文討論。

[5] 以上參考自梁伶君，「智慧卡簡介與應用趨勢」，成功大學圖書館館刊，第四期，1999/10，取自：http://www.lib.ncku.edu.tw/journal/journal/4/6_1.htm。

非對稱式加密方法是為了改善單以一把鑰匙加密狀況下，鑰匙必須傳送卻又可能在傳送途中遭竊的問題。非對稱式加密技術需要成對的鑰匙，每對鑰匙中任何人均可取得的那一把鑰匙，稱之為公開金鑰，另一把則是屬於個人擁有且必須妥善保管的，稱之為私密金鑰。使用公開金鑰加密的訊息，可用相對應的私密金鑰解密，而用私密金鑰加密的訊息，亦可用私密金鑰解密。例如甲要送訊息給乙，只要取得乙的公開金鑰後加密訊息，然後將加密後的訊息送給乙，乙再用自己的私密金鑰解密，就可以還原甲的訊息，如此一來，解密用的鑰匙不需要利用公開環境(如電話、郵件、網路等)傳遞，比起使用對稱式加密法，更能確保訊息的隱密。這部分的作法能確保文件僅能由事先設定好的對象解開閱讀。

非對稱式加密法另一項應用則是驗證發行者，如果乙收到的訊息，能用甲的公開金鑰還原時，表示該訊息一定是用甲的私密金鑰加密的，因此對乙來說，可以確保該訊息是由甲送出的，而且途中不可能經過任何人的更改，否則解密時就無法得正確的訊息。[6]但此種簡易的驗證方式，卻無法避免有心人士用另一金鑰對來偽造訊息。例如丙自製了一金鑰對，並偽稱自己是甲，然後用假金鑰對中的私密金鑰加密訊息並傳給乙，此時乙收到丙的訊息，並且正確地以公開金鑰解開，會以為丙真的就是甲。所以若要使用此方法來驗證身分，那麼乙用來解密的公開金鑰，就必須確認是甲的公開金鑰。所以在比較嚴謹的做法下，必須有公正的第三方來確認每一方的身分。但是在本研究的系統中，將這部分的做法簡化，上例中甲的公開金鑰就直接置於此驗證使用權的程式碼內並於程式一同編譯，由於驗證使用權的程式是由中央管理部門開發的，程式中讀取的是自己程式內的資料，因此在假定中斷程式碼與破壞程式完整性是困難的前提下，可以信任此程式。

由以上的討論中得知，若要將一份文件限制給特定對象閱讀，並且確認這文件是由特定發行者所發行的，就必須將這一份文件先以發行者的私密金鑰加密，再以要閱讀此文件特定對象的公開金鑰加密。

2.5 統一塑模語言：Unified Modeling Language(UML)

[6] 以上參考自查修傑，連麗真，陳雪美譯，「電子商務概論」，和碩科技，台北，1999。頁 100-107。原書為：Kalakota, Whinston, “Frontiers of Electronic Commerce”, Addison Wesley.

本研究使用在分析設計上的圖形表示法是採用 Unified Modeling Language(UML)，主要的原因是它專為物件導向分析設計方法而產生，並且支援整個系統發展流程中所需要的各種圖示，不像過去的結構化系統發展中所需的程式流程圖(Flow Chart)、資料流程圖(Data Flow Diagram)、資料庫個體關係圖(Entity Relation Diagram)等等，每一套都有自己的圖示表示法及隱含的系統發展概念。最重要的一點是，UML 已成為軟體工業的標準之一。[7]

UML 由 Rational 公司整合了物件導向分析設計方法的三位大師 Booch、Rumbaugh 與 Jacobson 的研究成果，自 1995 年 10 月的 UML 0.8 提交 Object-Oriented Programming Systems, Languages, and Applications (OOPSLA)，經過不斷地修訂及擴充，到 2003 年 6 月已由 Object Management Group(OMG)通過成為 UML 2.0。[8]

UML 包含九種模式圖：使用個案圖(Use Case Diagram)、類別圖(Class Diagram)、物件圖(Object Diagram)、循序圖(Sequence Diagram)、合作圖(Collaboration Diagram)、狀態圖(State Diagram)、活動圖(Activity Diagram)、元件圖(Component Diagram)、部署圖(Deployment Diagram)。由於本研究中只用到其中的 6 種模式圖，因此以下引用吳仁和《物件導向系統分析與設計：結合 MDA 與 UML》一書，簡略介紹這幾種圖形的功能。

1. 使用個案圖：是引用 Jacobson 方法中的使用個案模式，從使用者之觀點描述系統的行為者與系統間之互動行為與關係。從內部觀點來看，使用個案可描述系統做什麼(What)。從外部觀點來看，它可描述行為者與系統如何互動(How)。[9]
2. 類別圖：UML 之類別圖是引用 Booch 與 Rumbaugh 方法中的類別圖，主要用以表示系統存在之物件型態(或稱類別)及各物件型態間的靜態資料結構與邏輯關係，也表達類別之屬性、操作與類別間連結之限制等。
3. 循序圖：UML 之循序圖是結合 Booch 的互動圖與 Rumbaugh 的訊息追蹤圖而成，主要用以描述系統運作時物件間的互動行為，且著重以時間之先

[7] 參考自 Bernd Bruegge and Allen H. Dutoit, “Object-Oriented Software Engineering : Using UML, Patterns, and Java”, 2nd ed.(international), Pearson Prentice Hall, 2004。p.29. ”…because it provides a spectrum of notations for representing different aspects of a system and has been accepted as a standard notation in the industry.”

[8] 參考自吳仁和，《物件導向系統分析與設計：結合 MDA 與 UML》，智勝文化，台北，2005。頁 66-71

[9] 使用個案圖中的每一個使用個案代表一個系統完整的情境，包括其限制條件與該情境的活動流程。

後順序爲主軸，以表達物件間的訊息傳遞與處理程序。一個循序圖會有一個與之對應的合作圖，但表達的重點與方式不同。

4. 活動圖：UML 之活動圖可用於表達執行某一作業行爲中之活動、轉換與條件等。一個活動圖描述一群循序與同步的活動，一個活動可表示一個工作流程步驟或一個運算的執行動作。
5. 元件圖：UML 之元件圖起源於 Booch 的模組圖，用以說明系統設計過程各類別與物件的配置，以及途述軟體元件間的組織架構和關係。元件是開發和執行過程之實際物件的類別，將可分解的實際基本單位模組化，這些基本單位包括模組(Module)，並擁有特性和明確定義的介面。
6. 部署圖：UML 之部署圖起源於 Booch 的處理圖，它用來說明系統各軟硬體(例如處理器、處理元件)元件的配置、關聯，以及同一處理器內執行處理的時程安排等。

2.6 軟體系統發展方法：Rational Unified Process(RUP)

本研究探討軟體元件保護方法的需求探討及架構設計，並且將設計出一個可以實際運作的雛型架構，這個過程與軟體系統發展方法其實很類似，都必須了解領域問題、探討系統的需求、設計系統架構、完成系統實作等。而系統發展方法則具體提供每個步驟的指導原則與流程，以及分析設計用的工具(視覺化的標記工具，如 UML)。

RUP 是物件導向的系統發展方法，相較於傳統結構化的方法，RUP 提供了更完整的開發流程與指導原則，並且能夠從分析、設計輕易地對應到物件導向語言的程式碼上。而該流程結合了在整個開發過程中都能適用的 UML 標記法，讓整個程序更容易遵循。雖然物件導向的分析設計方法並不只 RUP 一種，但是 RUP 已融合了多位物件導向分析設計大師如 Booch、Jacobson、Rumbaugh 等多年的實務經驗與研究成果，因此本文採用此方法進行論文研究與論述。

以下簡略介紹本文所要使用的方法－RUP，其中四個階段與九項活動(六項工作流程與三項專案管理工作)的細節，將於下一章實際應用在本研究時做更詳細的介紹。

RUP 亦是由 Rational 公司所主導發展推出的標準，而其中的圖示標記法就是採用 UML。雖然 UML 提供了物件導向開發方式的標準圖示，但是並沒有提供一套完整的開發流程，RUP 正是補足了這一部分成為完整的軟體系統的開發方法。將這兩項技術(開

發流程與圖示標記法)分開，其實是一項刻意安排的結果，以便於相關的研究能聚焦且獨立進行。

RUP 的程序設計包括靜態與動態兩個維度，其中靜態維度描述的是九大活動，每一項活動皆有其活動的目標以及應該完成的產出(文件、模型、程式碼等)，而動態維度描述的則是四個主要專案階段，這四個階段是循序漸進，反覆執行，並且有明確的階段任務，而九大活動就是為了達成這些任務所要執行的工作。

這九大活動的內容略述如下[10]，前六項是完成專案必須進行的活動，後三項則是支援的管理工作：

1. 企業模型製作(Business Modeling)：這個活動是為了瞭解系統所要部署的目標組織之運作結構、整個企業的願景與環境，以及企業流程、角色、責任，以便在後續訂定專案目標時能符合企業願景且創造出更大的利潤。這個活動的產出是企業模型，以 UML 的使用個案圖、物件圖與活動圖來表示。
2. 需求導出(Requirements)：此活動是建立專案相關人員對於本專案該做什麼(What)與為何做(Why)的認同，並且描繪出該系統的界限，了解系統所應該達成的目標。最後的產出是需求文件，以 UML 的使用個案圖來表示。
3. 分析及設計(Analysis and Design)：此活動是將需求轉換成實作的系統設計，針對每一個需求的使用個案尋找可行的解決方案，最後的產出幾乎包含所有的 UML 圖形，但並非要求每一種圖形均完成，而是以能清楚表達設計為準，只要後續實作時能依圖就能完全了解設計即可。
4. 實作(Implementation)：此活動要實際將設計的結果轉成可運作的程式碼。最後的產出即是可執行程式。
5. 測試(Test)：此活動將實作出來的每個單元整合執行，確認是否可以運作無誤，並且驗證是否已達成系統規格的要求。
6. 配置(Deployment)：此活動是將完成的系統完整地配置在客戶端的環境中，包括包裝軟體、安裝測試、訓練使用者、轉換資料庫等。這部分的工作依企業環境等會有非常大的差異。
7. 組態管理與變更管理(Configuration and Change Management)：此活動的目的在

[10]參考自吳仁和，《物件導向系統分析與設計：結合 MDA 與 UML》，頁 49-51

追蹤與維護專案資產在演進過程之完整性。而每次反覆過程中的變更也必須詳細的記錄。

8. 專案管理(Project Management)：此活動是提供管理軟體專案的架構、實務準則，規劃專案生命週期及進展過程中的監督等。
9. 環境(Environment)：此活動的目的是以一些流程工具支援軟體開發，包括選擇與取得工具等等。

而 RUP 的四個階段也都有其明確的目標、里程碑，而每個階段所要進行的活動即是上述的九項活動中的某幾項甚或是全部。以下引用吳仁和《物件導向系統分析與設計：結合 MDA 與 UML》[11]的整理說明這四個階段：

1. 初始階段(Inception Phase)：主要建立可理解、完整的系統需求與範圍，建立接受準則與評估整體風險(包括成本、時程與技術等)，詳細構想出建構系統所需的企業個案，取得所有參與專案人員對推展該專案的認同。
2. 詳述階段(Elaboration Phase)：處理主要的技術工作(例如設計、實施、測試、系統結構等)，並以實際可執行的程式編輯來探討主要的技術風險(例如資源主張、績效與資料安全等風險)。
3. 建構階段(Construction Phase)：建構一個初步可運作的系統版本，繼續演化幾個內部版本(常稱爲α版)以確保系統是可用的及滿足使用者的需求。最後完成一個具有完整功能的系統版本(常稱爲β版)，這包括安裝與支援文件及訓練教材等。
4. 轉移階段(Transition Phase)：依使用者回饋精調系統、組態、安裝與可用性等議題，並將系統產品移交客戶使用，包括備份、培訓、支援、維護等。

RUP 結合了螺旋式的概念，以反覆與漸增的軟體發展概念來進行整個程序，因此「不需假設使用者的需求一開始就要清楚與完整地表達」[12]。RUP 包含動態與靜態兩個維度，以說明整個程序的階段與工作流程。如下圖所示。橫軸是四個順序性階段(由左而右)，但這些階段是可以不斷反覆執行的，每次的反覆都是從初始階段開始，而每個階段的執

[11]引自吳仁和，《物件導向系統分析與設計：結合 MDA 與 UML》，頁 47-48。

[12]引自吳仁和，《物件導向系統分析與設計：結合 MDA 與 UML》，頁 45-47。反覆發展指的是重覆用相同一系列的操作或步驟以進行軟體開發；漸增發展指的是每一次都在軟體產品上增加新功能、模組、子系統等。

行期間並不見得會一樣。縱軸則是六大工作流程與三種主要的專案管理工作。圖形的峰高則代表每項工作流程或管理工作在該階段的比重。

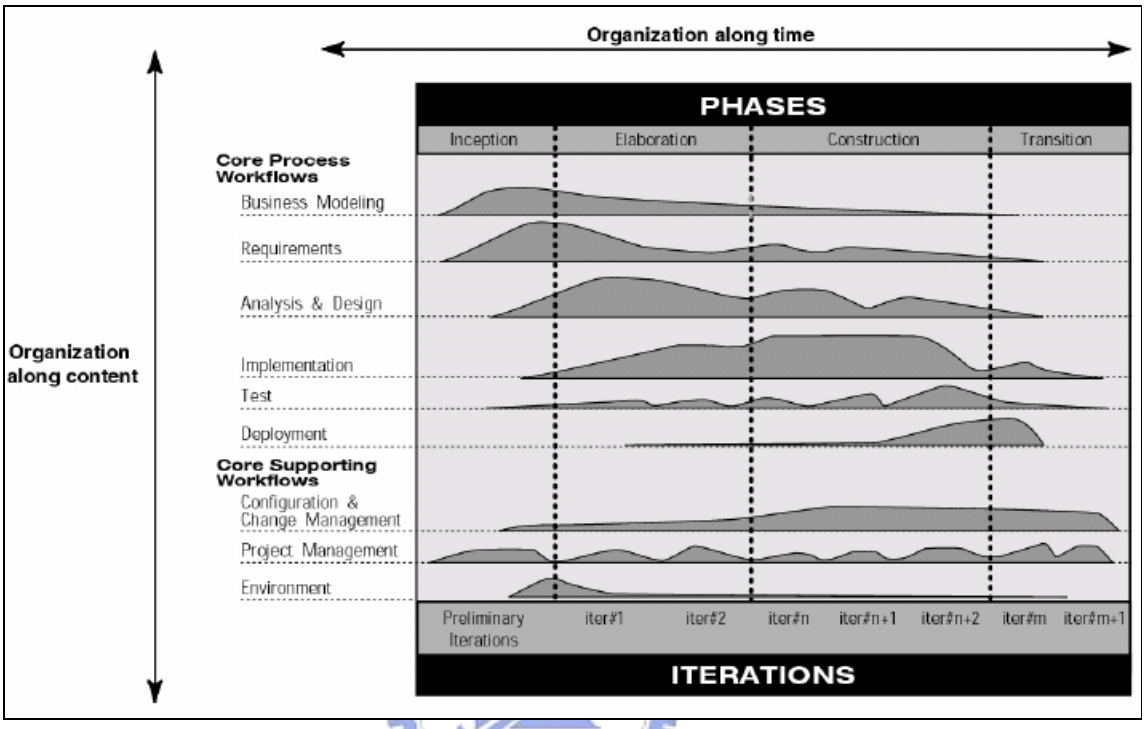

圖 2：RUP 的兩個維度

資料來源：Kruchten, "The Unified Process, An Introduction", p23

從上圖中階段與工作流程的對應可以發現，雖然四個階段所包含的工作項目會有許多項，但每個階段皆有其工作重點：初始階段－企業模型製作與需求分析、詳述階段－系統架構分析與解決方案設計、建構階段－系統實作、轉換階段－測試與配置。由於階段是依時序進行，因此本文亦將以這四個階段與工作重點的對應作為論述順序。

RUP 的程序雖然非常的完整，但並非要求任一系統或專案都必須完全按照其標準進行。如 Kruchten 所述「*A process should not be followed blindly, generating useless work and producing artifacts that are of little added value.*」[13]。RUP 提供的是一個架構與指導原則，實際使用時必須依照所要進行的專案大小、組織文化等進行調整。由於本研究僅是遵循其開發方法進行軟體保護方法的研究，並非要開發一項可散佈給客戶端的系統，因此在

[13] Philippe Kruchten, 《The Rational Unified Process, An Introduction》, 2nd Ed., Addison Wesley, 2000. p.31

六項工作流程中的配置(Deployment)並不會用到，而三項管理工作也非本研究的焦點，因此本文並不將這些工作的細節納入討論。

三、以 RUP 發展軟體元件保護方法

－ 以智光科技軟體公司為例

本文雖試圖發展一個共通的方法與架構以保護軟體元件不被惡意散佈，但保護方法的設計會視應用在不同的環境(如軟體公司、研究組織等等)以及組織對於元件既有的管理方法不同，而有實作時的差異。因此為了使本研究更容易聚焦以凸顯本保護方法的共通設計，本文選擇以一個虛擬的軟體公司(為求後文述敘方便，為此公司命名為「智光科技軟體公司」)為例來說明本方法的架構以及方法的設計理由。有關這個虛擬公司對於發展軟體元件的保護方法的需求，將在 3.1 小節中介紹。

而為智光科技發展這個保護方法的程序，正如 2.6 小節所述與發展一個軟體系統是相類似的，Rational Unified Process(RUP)有一套具體的程序指導如何從無到有發展一個資訊系統，因此本章將介紹為智光科技發展軟體保護方法時，在不同階段的工作目標、流程以及工作產出為何；下一章則實際依這些工作流程的項目完成每個工作產出，而對於元件保護的需求探討、方案設計、設計理由以及驗證設計的方法都會包含在這些產出之中。

3.1 案例說明－虛擬的軟體公司：智光科技軟體公司

智光科技軟體公司，專門製作高價值的元件，如高效率的加解密元件、壓縮與解壓縮元件等，並且銷售給企業或個人使用，公司內有一研發部門與測試部門，研發部門利用他們專業的知識，研究並設計這些高價值的元件的演算法，並且將這些演算法實作成元件，然後交給測試部門進行測試評估；在未完成測試之前，為避免這些元件外流出去，所以要保護這些元件，因此希望在這些元件加上使用控管機制，使元件只能在測試部門的電腦上使用，避免這些元件被竊取，或者是由測試人員蓄意帶出公司散佈。未來元件上市的時候，公司可運用同樣的機制，讓購買元件的企業或消費者，只能在特定的電腦上使用，避免消費者將元件散佈到市面上，損傷公司的利益。

因此智光科技期望能開發出一套支援元件保護功能的方法，以達成上述的目的。

3.2 反覆(Iterative)式的發展流程

RUP 的四個發展階段看起來是循序漸進，但並非到了第四階段結束就完成整個專案，而是可能會再回到第一階段修正專案範圍與工作，反覆進行到完成並確認專案目標後停止。本研究進行的方法一樣是採反覆式的流程，首先探討軟體元件保護的目標與相關研究，直到設計出雛型架構，再驗證是否達成本研究的目標，然後繼續進行下一次的反覆，做出更明確的設計。第一次的反覆過程中，研究目標是模糊的，直到幾次尋找設計解決方案、嘗試實作測試後，再從第二次反覆的初始階段開始，漸漸釐清本研究確切的目標與解決方案，並且設計每個類別的細節(例如屬性、方法、參數等)。但為使本文的閱讀更為流暢，以下直接以這四個階段的順序做為書寫順序，需要注意的是，每個階段的成果實際上可能是經過幾次的反覆過程後的最終成果。

3.3 RUP 四個階段與發展智光科技元件保護方法程序的對應

RUP 的四個階段分別是初始階段(Inception)、詳述階段(Elaboration)、建構階段(Construction)、轉換階段(Transition)。在這四個階段中，原本是需要進行多項活動的，但是由 2.6 小節圖 2 的說明中可知每個活動最大的工作量其實可以鎖定在某一個階段中，因此以下說明本研究在這四個階段中所應該完成的目標、在該階段所應該進行的活動、以及每項活動所該完成的產出，其中僅第一階段包含兩項主要的工作，其他階段的主要工作都只有一項。在 Kruchten 一書中，每個活動都包含了必須進行的工作項目說明，並且以 UML 的活動圖繪出該流程的進行步驟。而智光科技為發展元件保護方法，可視為公司的一項專案，因此下文將以本專案指稱發展元件保護方法這件事。本章除了介紹 RUP 的具體程序之外，尚會說明每個活動都會在本專案中該進行的對應工作，並且繪出工作項目的活動圖，下一章實作的內容即是根據本節每個階段每項活動完成後的產出。

3.3.1 初始階段(Inception)：企業模型製作

目標：本階段的意義在於目的企業的結構、資源與未來願景等，做為後續規劃的指導方針。

活動：此階段的主要活動是完成企業模型製作。企業模型製作在於了解公司的組織

結構與營運方式，對於製作的詳細程度則視專案的目標與預算、時間等而有非常大的不同。在本專案中僅著眼於跟元件保護直接相關的企業模型。

企業模型製作流程：本工作流程包含以下幾個重要項目[14]。當專案的目標僅是為企業完成一個特定功能的系統時，這個活動要做的事情是很少的。[15]

1. 定位企業的狀態：這項工作在於了解企業的願景。在本專案中，智光科技對於系統的需求已是非常明確，因此不需在這工作上花太多時間。
2. 製作領域模型：這項工作在於完成某個特定領域的模型。以本專案而言，就是了解智光科技在元件保護上的問題，為專案聚焦。因此製作的模型均是與元件保護相關的。在 3.1 節的情境說明中，只有元件開發與測試流程是與本專案直接相關，因此會了解目前流程與已存在的問題。

產出：

1. 智光科技的組織架構。
2. 智光科技元件開發與測試流程。

3.3.2 初始階段(Inception)：需求導出

目標：本階段的意義在於定義整個專案的目的，劃定專案範圍，規劃專案時程等，以描述智光科技對於元件保護的需求。

活動：此階段的主要活動是完成需求導出。需求導出則是實際依公司狀況，衡量預算與資源，導出希望在這次專案所達成的目標。

需求導出流程：本工作流程包含以下幾個重要項目[16]，其活動圖如圖 3 所示。

1. 分析領域問題：這項工作是將要解決的問題文件化，並且定義出系統的邊界。在本專案中即是了解元件保護的對於智光科技的意義以及所扮演的角色。

[14] 參考自 Philippe Kruchten, 《The Rational Unified Process, An Introduction》, p.146-148。
[15] 如果專案進行的目標，並不只是完成一個特定目的的系統，而是希望整體地了解企業體質與未來發展，那麼這個活動的工作就顯得格外重要，並且有許多工作項目要逐一完成。
[16] 參考自 Philippe Kruchten, 《The Rational Unified Process, An Introduction》, p.163-165。

2. 了解業主的需求：這項工作是要了解使用者對系統的需求，以及業主希望該系統能為企業帶來什麼樣的效益。在本專案中即是了解智光科技對於元件保護有何需求與期待。
3. 定義系統：這項工作是從上述的需求中，整理出系統該具備的特徵以及限制。在本專案中即是定義元件保護系統該具備哪些特性以符合智光科技的需求。
4. 管理系統視界：這項工作是收集需求的相關資訊，以客戶的期待規劃專案期間、預算等。在本專案中即是了解智光科技的資源與規劃本專案的細節。
5. 重新修訂系統定義：在管理系統視界的時候，有可能發現系統的其他特徵，因此可能會回到流程 3 修訂系統定義以及將限制更明確地記錄下來。這項流程就是一個小單元的反覆過程，最終的產出將整合於步驟 3 之中。
6. 管理改變的需求：當系統發展過程中，業主可能會隨時有新的需求希望這個專案能達成，因此必須將這些改變納入管理，審視是否會違背原先的目標，以及該如何進行以達成這些改變。

產出：

1. 智光科技對於元件保護的需求分析文件。
2. 元件保護系統的定義(能做什麼及達成什麼目標)。
3. 以使用者觀點繪製的系統使用個案圖。

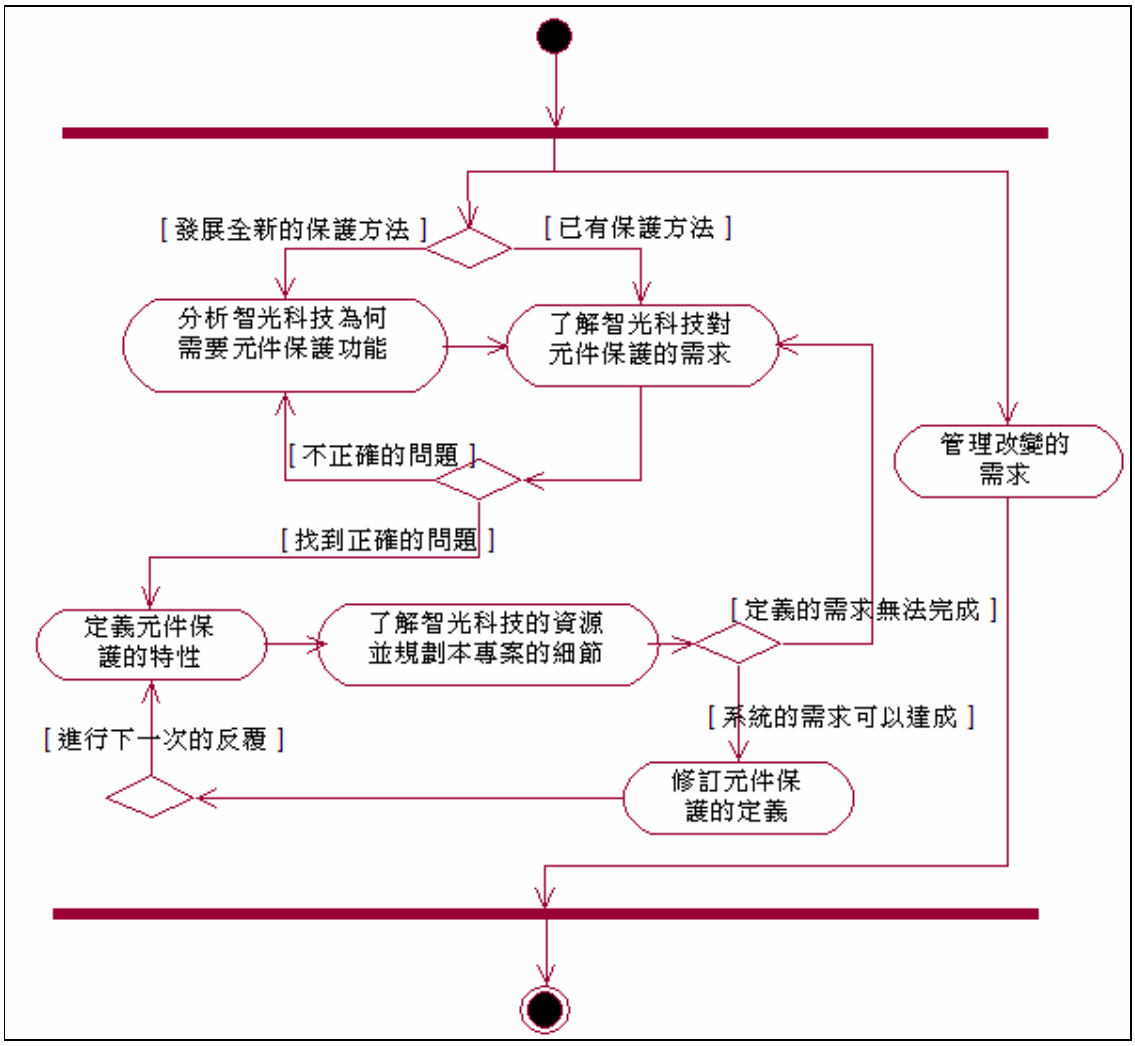

圖 3：RUP 初始階段-需求導出流程(UML 活動圖)

參考自 Kruchten, 《The Rational Unified Process, An Introduction》, p.164

3.3.3 詳述階段(Elaboration)：分析與設計

目標：本階段的意義在於為初始階段的需求與目標做更進一步的描述，使開發人員能夠完全的了解系統的需求及功能，讓下一階段建構系統時有明確的遵循規格。

活動：此階段的主要活動是分析與設計。對本專案而言，就是要找出適用於智光科技的軟體元件保護解決方案，然後評估這些解決方案是否能達成系統目標，並轉換成實作時可參考的模型圖。

流程：分析與設計的工作流程包含以下幾個重要項目[17]，其活動圖如圖 4 所示。

1. 定義一個可用的架構：這項工作在剛開始的反覆時，會先找出可用的架構，後續的工作就可以依據這個架構進行分析以及選擇解決方案。對本專案而言，即是繪出適用於智光科技元件保護系統的架構以滿足初始階段－需求導出中的使用個案。
2. 重新修訂架構：架構在進行分析時是可能會不斷演變的，但每一次的演變均必須確認與系統目標一致。
3. 分析系統行為：這項工作是依據以上的架構，分析出系統內部應該如何運作。對本專案而言，即是以更詳細的使用個案，描述元件保護系統該具備哪些功能，以及這些功能的關係。
4. 設計元件：由於 RUP 的設計觀念是以元件為出發的，因此系統的功能是以元件來達成。對本專案而言，即是針對每個使用個案設計出能滿足這些系統行為的元件。
5. 設計即時反應的元件：系統有一些是需要即時反應的功能，這些部分的元件設計可能有多執行緒的設計，因此略不同於流程 4 的一般性元件。由於本研究的重心在探討整個元件保護的需求與架構，因此暫不考慮這個部分。
6. 設計資料庫：這項工作對於有大量資料需儲存的系統才需要執行，這部分的工作在於找出哪些類別是需要永續儲存的，然後依據採用的資料庫類型定義相對應的結構。在本研究中，將重點放在驗證的過程，而且資料庫的結構會因組織結構而有相當大的差異，因此本文不實作這部分的設計。

產出：

1. 更明確的系統功能需求與解決方案。
2. 從系統行為分析的使用個案圖、活動圖。
3. 使用個案所轉換的類別圖與循序圖。
4. 元件保護的佈署架構與元件圖。

[17] 參考自 Philippe Kruchten, 《The Rational Unified Process, An Introduction》, p.177-181

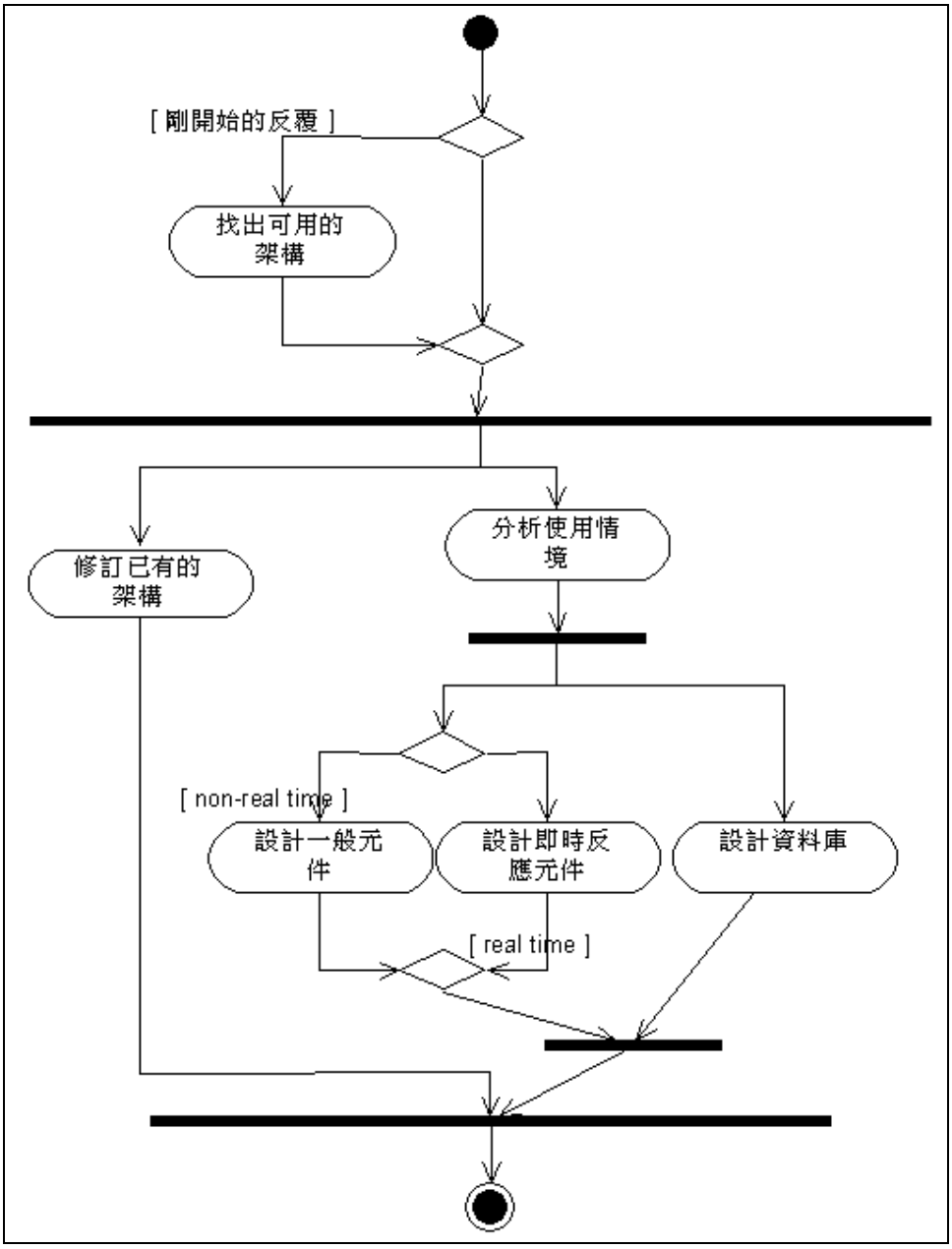

圖 4：RUP 詳述階段-分析設計流程(UML 活動圖)

參考自 Kruchten, 《The Rational Unified Process, An Introduction》, p.178

3.3.4 建構階段(Construction)：實作系統

目標：本階段的意義在於將詳述階段的模型實際對應成程式碼，並且將每個子系統都整合成完整的系統。在這個階段中也會進行系統的測試，但重點是開發人員的內部測

試，確認是否已完成系統規格上的需求。

活動：此階段的主要活動就是實作。對本研究而言就是要將整個系統雛型實作完成，並且做完單元與整合測試。本文的重點在於元件執行時期的控管，因此與這程序相關的使用個案均會實作出來，而其他支援性的管理工作(如組織階層與機器的管理)與本研究重點較無關，而且這些功能在實作上較無困難，由於時間限制，故只有選擇性的實作，但仍然以元件執行時期能正常運作為首要目標。

流程：實作的工作流程包含以下幾個重要項目[18]，其活動圖如圖 5 所示。

1. 確認實作模型已完成：實作模型其實是詳述階段時就該完成的工作，這裡只是再確認這些實作模型都很完整，而且模型的語意都很清晰。
2. 計劃功能整合：由於系統功能是由許多元件組合而成的，所以在這工作項目中，必須安排好元件要如何整合，使得子系統可以正常運作，這也會影響元件的實作順序。不過在本專案中，關鍵的元件與功能並不多，佈署也不複雜，因此可以容易地進行整合。
3. 實作元件：當上述的計劃完成時，就會依照計劃順序實作元件。實作元件時尚須完成單元測試，確定該元件可以正常運作無誤。
4. 整合子系統：當一個子系統所用到的元件均實作完成並經過單元測試後就可以進行子系統的整合測試。
5. 整合系統運作：當子系統都經過測試後，就會整合成完整的系統。以上流程 3-5 並非依序做完一次就好，而是一個反覆的過程，直到每一個元件、子系統均完成，且通過完整的整合後才會結束。

產出：

1. 元件可執行檔。
2. 元件執行時相關的設定檔案。
3. 元件保護方法的支援網站。[19]

[18] 參考自 Philippe Kruchten, 《The Rational Unified Process, An Introduction》, p.189-192

[19] 本架構會有一個中央伺服器來接受元件申請並產生元件授權書。詳細設計請見本文第四章。

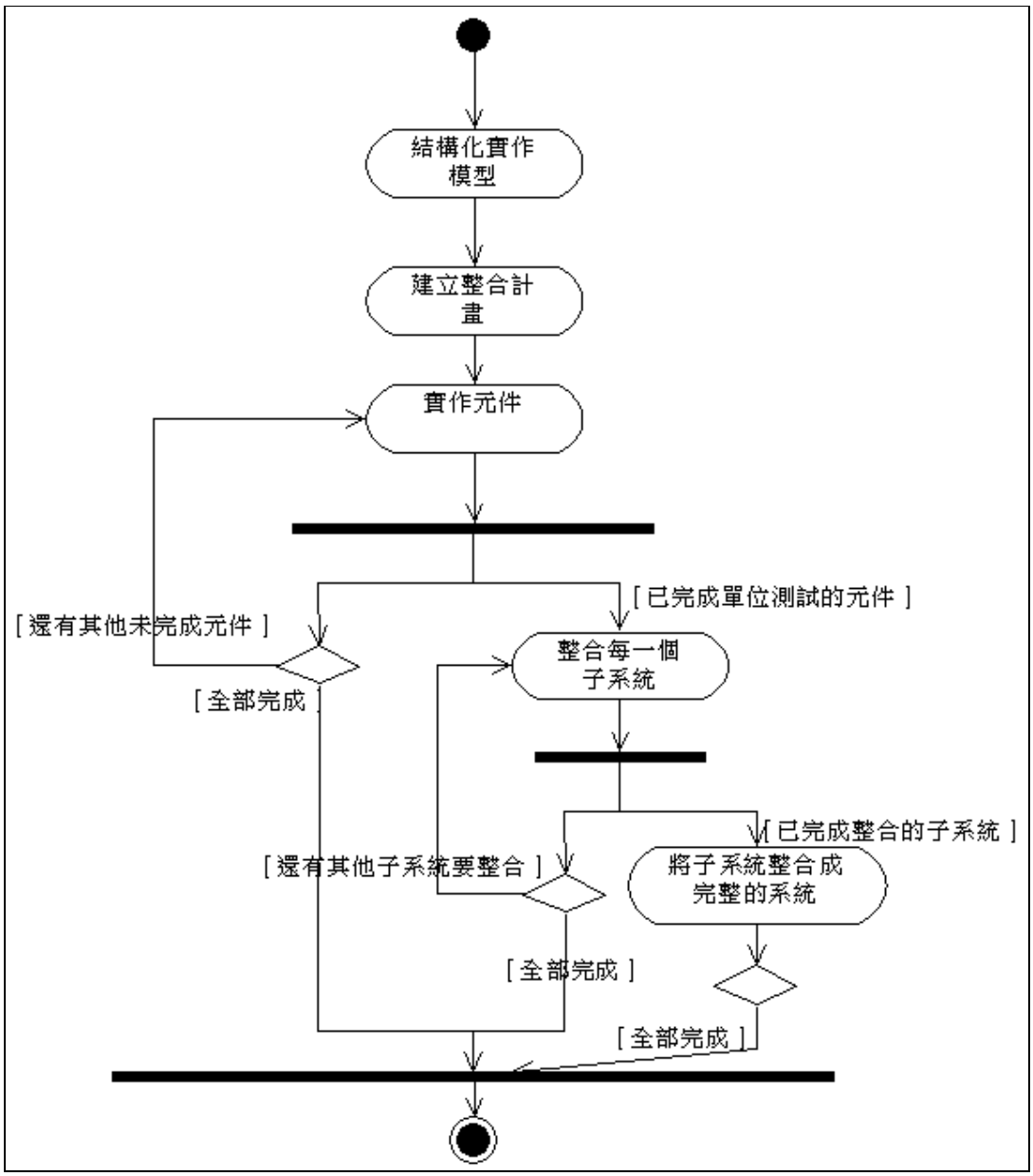

圖 5：RUP 建構階段-實作流程(UML 活動圖)

參考自 Kruchten, 《The Rational Unified Process, An Introduction》, p.190

3.3.5 轉換階段(Transition)：測試系統

目標：本階段的意義在於將實作完成的系統佈署到客戶端，並完成最終的使用者測試與教育訓練。

活動：此階段的主要活動是測試，雖然實作時就已經將整個系統整合在一起並測試

過，但實作時測試的重點是確認系統規格均已實作無誤。而此階段的重點則是由使用者由外部測試是否達成原先的預期。本專案在此階段會以模擬使用情境來進行整個流程的測試。

流程：分析與設計的工作流程包含以下幾個重要項目[20]，其活動圖如圖 6 所示。

1. 規劃測試工作：這項流程是擬定有哪些項目要測試，以及測試流程。
2. 設計測試方式：這項流程則是依要測試的項目設計測試的方式。在本專案中，會設計元件在正常與不正常使用的模擬情境來進行測試，以確認元件是否受到預期的保護。
3. 實作測試程式碼：有些測試工作必須進行多次，所以可能會撰寫程式碼來簡化測試工作。在本專案中，將會實作一個簡單的應用程式來引用受保護元件。
4. 整合測試：這個項目是確定每個子系統整合的部分都有正確的連結。
5. 系統完整測試：確定整個系統功能如原先的設計，並且滿足使用者的需求。本專案在此流程時會執行設計好的測試應用程式，確認在所設計的不同模擬情境中是否能正確運作。
6. 評估測試：測試工作完成後，必須評估測試的結果是否如預期般，若沒有，則必須將問題記錄供下一次反覆時改善。

產出：

1. 測試用模擬情境。
2. 測試用需受保護的元件。
3. 測試用應用程式。
4. 測試結果。

[20] 參考自 Philippe Kruchten, 《The Rational Unified Process, An Introduction》, p.202-205

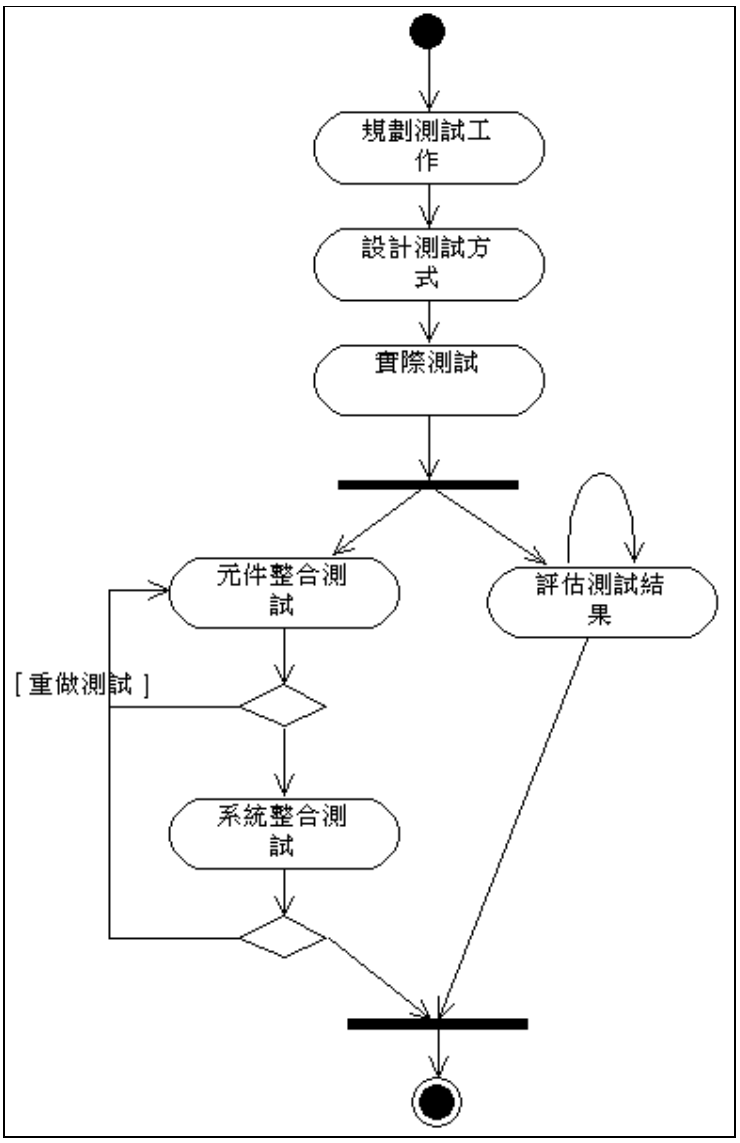

圖 6：RUP 轉換階段-測試流程(UML 活動圖)

參考自 Kruchten, 《The Rational Unified Process, An Introduction》, p.190

3.4 討論

RUP 的程序可以具體地指導如何完成一個系統的開發，並明確指出系統開發該有的產出。但是一般的系統開發在選擇設計方案時，通常只會將最後選擇的方案畫成設計模型圖或是說明文件，並不見得會包含為何選擇此方案的理由。而且發展系統的產出有不同的性質，如文字文件、模型圖、程式檔等等，這些文件可能會分類集結成不同的類別而交付客戶。但在本文中，為了能清楚表達解決問題的過程，因此將這些系統發展的產出以發展階段重新組織，並且將一些方案的討論列入產出的說明之中(如使用個案說明

表格的其他說明一項)，以呈現研究的歷程。

四、軟體元件保護方法探討與系統雛型建置

- 以智光科技軟體公司為例

本章將依循第三章所說明的步驟，探討在智光科技公司的環境中，對於元件保護有何需求，然後尋求解決方案並且實作出系統雛型，以驗證是否達到元件保護的目的。依本文 3.3 節的四個階段與對應的工作流程順序進行會得到特定的產出，這些產出即是本研究的研究成果。

4.1 初始階段：企業模型製作

本專案的目的在於為智光科技開發元件保護系統，而非重新規劃公司整體營運，因此只需要了解公司與元件保護相關的資訊即可。由 3.1 節所說明的情境可知，本系統會牽涉到公司的不同部門與元件開發測試流程，因此會以一公司的組織圖來表示這些部門的關係，以及用 UML 活動圖表示公司在元件開發與測試的流程。

本節依據 3.3.1 小節的流程，產生公司組織圖、研發部開發元件與測試部測試元件的活動圖。以下摘要其結果及說明。

4.1.1 智光科技的組織架構

智光科技是一個小型的軟體公司，因此部門設立較為簡單，在總經理室之下，有會計部、研發部、測試部、業務部、員工福利委員會。本專案直接相關的部門則是只有研發部與測試部。其組織圖如下所示：

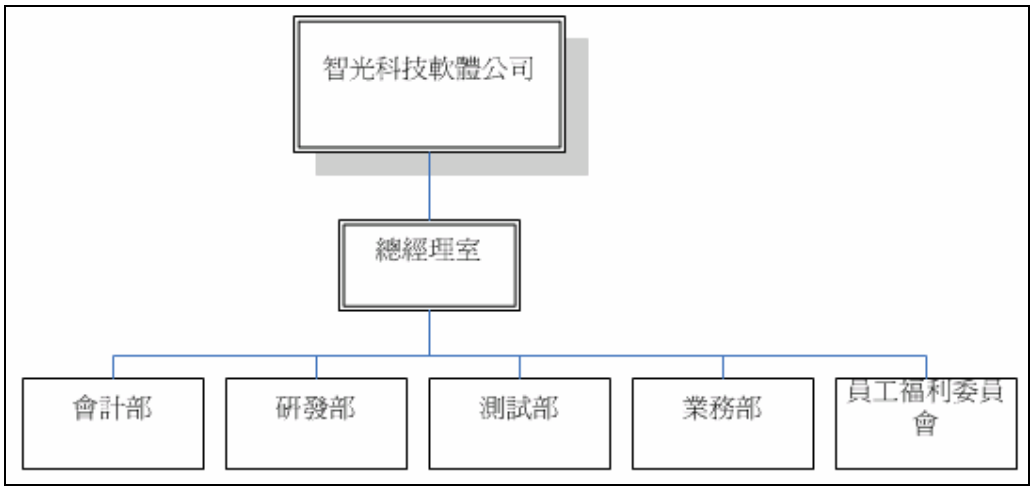

圖 7：智光科技軟體公司部門組織圖

4.1.2 智光科技元件開發與測試流程

公司會依據目前市場上的需求以及公司內人員的專長，設立不同的專案開發新元件或是為已有的元件改版。當研發部完成一個全新的元件或是新版的元件時，就會遞送給測試部要求評估該元件的功能與效能，測試部再將這些測試結果回報給研發部參考。該公司的策略是即時要開發新版元件，亦當成開發全新元件一般，甚至會交給不同的小組負責，以激勵出新的設計方法。整個元件開發的流程如下圖所示：

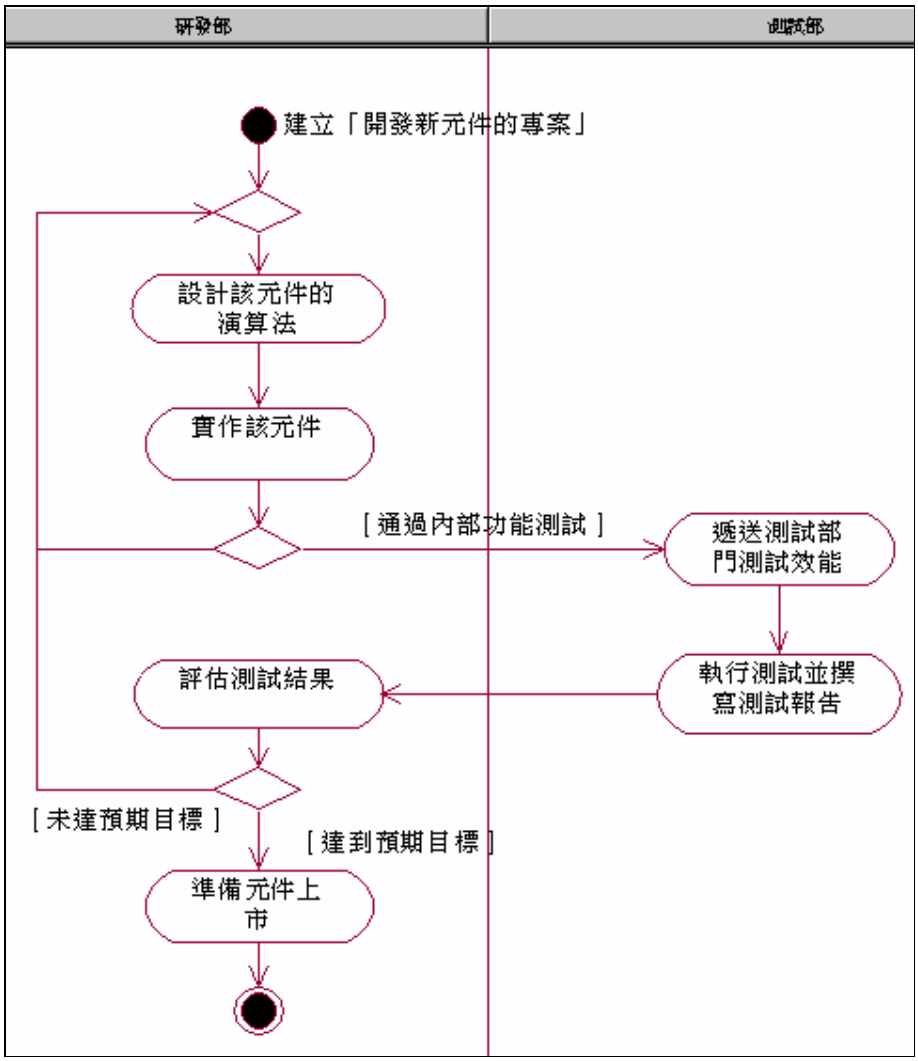

圖 8：智光科技-企業模型製作-開發與測試流程(UML 活動圖)

4.2 初始階段：需求導出

本節依據 3.3.2 小節的流程，產生需求說明、元件保護系統的定義、使用個案圖。以下摘要其結果及說明。

4.2.1 分析智光科技對於元件保護的需求

由 3.1 節所說明的情境中，智光科技對於本專案的目標非常明確，就是當元件在測試階段時，僅能在測試部的電腦上執行；如果元件被複製到其他非測試部的電腦上，則會停止其運作。而這樣的機制在未來上市的時候，能夠輕易修改即應用在消費者上。

公司研發人員的專長與工作應該集中在能為公司創造利潤的元件功能上，如果研發人員為了在元件加入使用控管機制，而必須大幅度的改變開發方法或是增加額外的開發設定工作，會減損原先的生產力。故希望該系統設計能盡量減少開發人員的額外工作。

而智光科技所銷售的元件未來必定是給其他應用程式(其他公司購買此元件，應用在該公司的資訊系統上)使用，但是卻不能大幅度的改變其他公司開發應用程式的習慣，因此雖然元件會限定在特定的電腦上使用，但是要有自行驗證是否在已授權的機器上執行的能力，這樣使用該元件的應用程式就不需有大幅度的更動。

4.2.2 定義元件保護系統

由上一節的需求說明中，我們可以整理出這個系統應該具備的兩個主要特性：

1. 元件開發人員能「輕易地」為元件加上使用控管機制，限定只能在特定的電腦上使用。
2. 元件具備自我檢查機制，引用的應用程式只要有合法授權，就不需了解元件檢查的機制。

4.2.3 以使用者觀點繪製的系統使用個案圖

從上一節的定義，我們可以將智光科技對於這個系統的需求繪製 UML 的使用個案圖。而在初始階段時使用個案圖的意義在於表達使用者對於系統的預期，忽略系統的實作設計，這就是所謂的使用者觀點。上一節定義中的第一項「為元件加入使用控管機制」為功能性需求，所以會以一個使用個案[21]來表示；第二項是屬於這一個使用個案的條件，因此在同一個使用個案中描述。繪出的使用個案圖[22]如下：

[21] 在 UML 的使用個案圖中，人形符號代表「參與者」(Actor)，橢圓形符號代表「使用個案」(Use Case)，帶有箭號的線是任兩個參與者與使用個案的關係。詳細的介紹請參見其他 UML 的書籍。

[22] 使用個案可以在系統開發的不同階段中使用，在一開始使用個案是很簡單的，到愈後來的階段，其設計細節愈明確之後，可能會變得愈來愈複雜，有的使用個案亦會分解成多個。在這個階段的使用個案圖僅用於表示系統主要的需求，可以說是「概念模型」。

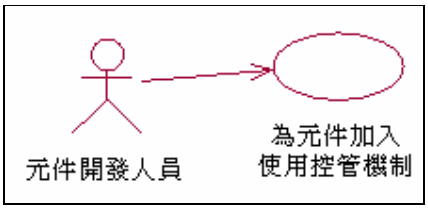

圖 9：智光科技-需求分析-使用個案圖(UML 使用個案圖)

由於該使用個案有一項限制條件需要加以說明[23]，因此列表如下：

表 2：使用個案說明 - 為元件加入使用控管機制

名稱：	為元件加入使用控管機制
啟動者：	元件開發人員
限制條件：	當為元件加入這個機制時，必須將程序簡化，並提供工具自動化。
流程說明：	1. 開發人員將元件開發完成。 2. 開發人員依照系統提供的流程與工具，為元件加入控管機制。

4.3 詳述階段：分析與設計

本節依據 3.3.3 小節的流程，產生更明確的元件保護系統需求分析、使用個案、元件佈署架構與元件圖。其中使用個案會包括其說明、活動圖、類別圖、循序圖，這些設計的結果必需提供足夠的資訊讓建構階段的實作可以依循。以下摘要其結果及說明。

4.3.1 明確化定義元件保護系統功能需求

本小節從系統功能面探討此系統應該具備哪些功能，而這些功能都必須符合需求導出階段的系統定義。這部分的功能定義是反覆進行現有方法探討、尋找解決方案、設計解決方案、評估解決方案後產生的。

為了達成本系統的目的，本文首先收集了現有的軟體保護方法，探討這些方法是否適合應用在元件上及其優缺點。由於現有方法的介紹與比較在 2.2 節中已說明，在此不

[23] 使用個案的說明在於表達一個使用的情境與流程，如果使用個案非常容易理解的時候，並不見得需要加上說明表格。當使用個案的流程較為複雜時，會視需要搭配活動圖來表示。

贅述。在 2.2 節的最後，已經整理出現有方法的缺點，因此在本系統的設計時，必須能夠解決這些缺點，所以改善這些缺點的方法就變成了本系統不可或許的功能，再加上之前的需求分析，將這些功能需求整理於後：

1. 元件被呼叫時，會啟動自身的使用控管機制。(2.2 節缺點整理第 1 項)
2. 元件僅能在已限定的電腦上執行。(4.2.2 節的定義第 1 項)
3. 元件的控管程序需獨立於元件外，以適合大量的元件，且能「輕易」與元件結合。(2.2 節缺點整理第 5 項、4.2.2 節的定義)
4. 元件的使用控管資訊需容易製作，但卻不容易被複製。(2.2 節缺點整理第 4 項)
5. 如果執行元件的電腦擁有正確的授權，應用程式的執行不須介入此元件的檢查機制，以期與使用未加入使用控管的元件一樣。(4.2.2 節的定義)
6. 元件的使用控管機制可以在單機上完成，不需仰賴網路。(2.2 節缺點整理第 2 項、第 3 項)

4.3.2 探討系統功能的解決方案

接下來必須探討如何滿足上一節所定義的功能需求，以下將逐一說明這些需求的解決方案。

1. 在元件啟動時檢查使用權：由於每個元件都有其進入點，因此元件的開發人員必須在這個進入點加入檢查使用權的程式碼，如此一來不管元件被竊取或是惡意散佈，這個檢查程序都會跟著元件，也保證元件使用時都會檢查使用權。[24]
2. 使用元件授權書記錄使用權允許清單：由於元件被限定只能在特定的電腦上執行，但是當元件上市的時候，元件所限定的電腦就不再只是公司內的測試部，因此為了提供彈性，必須設計能夠輕易變更限定對象的機制。所以必須將限定對象的資訊獨立於元件之外，配合第 6 項的需求，檢查工作必須在單機上完成，所以將限定對象的資訊製作成「元件授權書」隨著元件一起散佈到已限定的電腦上即可。由於元件依存於元件授權書，因此非法使用者若是只竊取元件到自己的電腦上，缺乏元件授權書並無法執行元件；即使將授權書一併複製到自己

[24] 如果是利用光碟防複製技術或是一次安裝方案，一但該元件安裝於擁有使用權限的 A 電腦後，就可以輕易地複製到 B 電腦上使用。詳細討論請見第 6 頁：林祝興，李鎮宇，「網路軟體保護方法之研究：一次安裝方案」。

的電腦上，但授權書上記錄的電腦資訊與正在執行中的電腦資訊不符合仍然無法執行該元件。

3. 實作使用控管代理人：第 2 項的需求就是要將元件的檢查程式獨立於元件之外，這樣一來，不同的元件不需個別撰寫檢查的程式碼，元件開發人員在元件啟動的程式碼也只要將檢查工作委派給「使用控管代理人」即可，而且委派的程序是標準化的，如此開發人員可以很容易就將這機制加入元件。(詳細的差異可見下文的實作程式碼)
4. 將元件授權書製作成檔案：由於元件控管資訊記錄在元件授權書內，如果擁有正確的元件授權書就能使用該元件。而元件授權書如果實作成硬 keypro 或是光碟母片，都較為容易製作，的確可以不容易被複製，但是其製作成本高且使用時需要抽換而造成不便；相對於檔案，可以很容易的由軟體產生，對於大量的不同元件並不會有製作上的困難，而且檔案放置於目標電腦的硬碟上，不容易有空間不足的問題(如智慧卡就會有空間限制)。但問題就是要避免授權書一併被帶到目標電腦上，因此後續針對元件授權書有加強的設計，使得授權書雖然可能被非法使用者複製，但是卻無法使用。
5. 元件在透過使用控管代理人檢查之後，如果使用權檢查無誤，則會繼續執行應用程式所要求的功能，不會產生任何訊息，因此應用程式並不知道該元件已經執行過檢查程序；但如果說目標電腦未擁有使用權，則元件會停止執行功能而產生錯誤訊息，而捕捉錯誤訊息本來就是應用程式的工作，因此並不會影響應用程式的開發與使用方式。[25]
6. 第 6 項的需求由於元件授權書已與元件一起散佈到目標電腦上，因此檢查時不需透過網路。

以上的功能分析是為了達成在 4.3.1 小節裡的系統需求，並提出可行的解決方案。由於分析設計是不斷反覆的進行，當更深入探討某一項功能時會再修正、改變設計方式，或是為加強某項設計而增加其他的功能。這些細節的分析與設計，將在下一節的使用個案中逐項討論並說明於使用個案的「其他說明」一項。

4.3.3 從系統功能分析的使用個案

[25] 應用程式在引用元件的時候，本來就可能產生其他類型的錯誤，例如系統權限錯誤、找不到元件等，所以當應用程式引用元件的功能時，必須自行捕捉處理可能發生的錯誤。

由以上的功能需求與解決方案說明，可以得知主要的系統功能為：製作元件授權書、為授權書加入防複製功能、製作受保護元件、請求「使用控管代理人」進行使用權檢查、「使用控管代理人」控管使用權。這些功能之間有依存關係，在使用個案圖中以視覺化圖形表示會更為清楚(圖 10)。與這些使用個案相關的「參與者」除了上述的「元件開發人員」這樣的自然人之外，「受保護的元件」會啟動其中的使用個案、「應用程式」引用「受保護元件」，都算是使用個案的參與者，其中「應用程式」不在本系統的邊界[26]內。

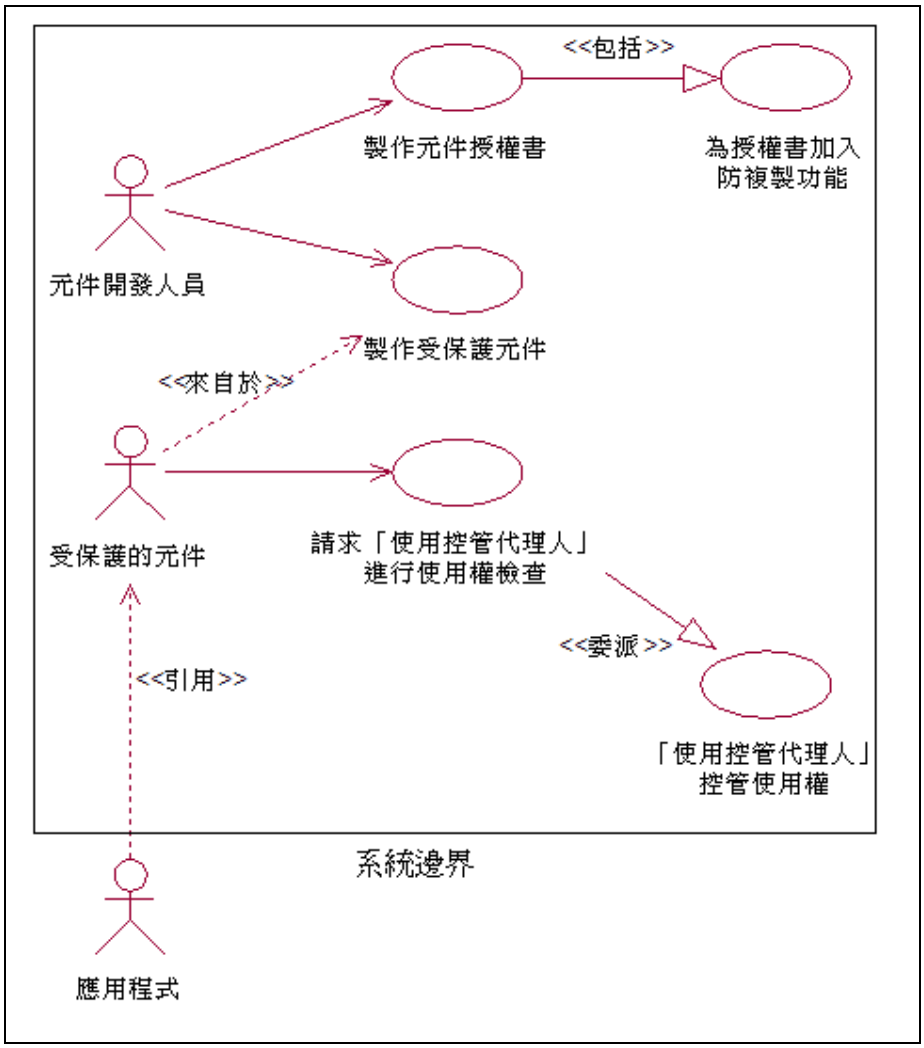

圖 10：智光科技-分析設計 - 系統功能使用個案圖

系統的使用個案說明於後以表格列示(表 3 到表 7)。

[26]由於系統可能會與外部系統溝通，因此在使用個案圖中用系統邊界來凸顯系統範圍。

表 3：使用個案說明 – 製作元件授權書

名稱：	製作元件授權書
啟動者：	元件開發人員
限制條件：	必須選定與何元件共同使用、以及欲使用該元件的電腦識別資訊
其他說明：	1. 以元件名稱、類型、版本、檔案名形成每一個元件的唯一識別碼，再將此識別碼存於授權書中。如此一來便不能以 B 元件的授權書偽裝成 A 元件的授權書來通過驗證程序。 2. 在前述的需求中，元件授權書會有允許清單，但是檢查使用權之時只要確定該電腦是否在清單內，而製作授權書時若已指明該授權書將於何電腦上使用，則授權書只要記錄該電腦的資訊即可。[27] 因此即使同一元件其元件授權書亦不同。 3. 由於智光科技是專門設計元件銷售給客戶，因此公司內會有許多的元件，這些元件通常會集中管理，為配合公司系統的資源與知識管理發展，因此將這個使用個案的功能以網站的形式實作，但網站其他的支援性功能則不在本研究討論範圍，因此忽略之。
流程說明：	1. 開發人員選擇要製作授權書的元件。 2. 開發人員指定要將該元件散佈到哪個特定電腦上。 3. 網站伺服器讀取開發人員所指定的元件資訊與電腦資訊。 4. 網站伺服器檢驗所指定的元件是否可在指定的電腦上執行。若無則拒絕製作授權書。 5. 網站伺服器將授權書的基本資訊(元件資訊+電腦資訊)準備好後，啟動「為授權書加入防複製功能」。 6. 將製作好的授權書給開發人員下載。

表 4：使用個案說明 – 為授權書加入防複製功能

名稱：	為授權書加入防複製功能
啟動者：	製作元件授權書

[27] 電腦資訊的選擇可以有很多種，唯一的要求就是必須能證明該機器的身分，但必須是不容易被竄改或偽造的資訊。例如微軟公司的 Windows XP 即以主機板的 BIOS ID、硬碟廠牌容量、顯示卡、網路卡等資訊進行雜湊運算，以識別不同的機器。本文將電腦資訊簡化成網路卡卡號與 IP 位址，實作應用則可因組織環境與安全性等級的要求而選擇不同的電腦資訊。

限制條件：	已準備好的原始授權書。(元件資訊+電腦資訊)
其他說明：	1. 由於授權書是以檔案形式存在，因此如果不針對授權書加以保護，則非法使用者可以自行竄改授權書內容，使得保護的目的失效。而保障內容不被竄改的方法，可以模仿目前已成熟的數位簽章方法。公司會為使用到受保護元件的電腦製作非對稱式加密法的金鑰對，在目前就是測試部的電腦，但未來若將元件銷售給其他消費者，亦可為消費者產製金鑰對，公開金鑰儲存於公司的網站伺服器上，私密金鑰則透過安全方式傳送到該電腦上。其原理與方法請參見本文 2.4 節。 2. 當使用上述的簽章方法時，電腦的私密金鑰需經嚴密保護。否則私密金鑰可能經由合法使用者提供給不合法的電腦使用，因此必須將私密金鑰保護不被複製。私密金鑰如果單純以檔案儲存在硬碟、軟碟片、光碟等媒體上，非常容易被複製，即使是防拷貝的技術也都逐一被破解；雖然亦可以實作成硬體以儲存私密金鑰，但如果是特製的硬體其製造成本高，其存取的方法亦缺乏標準化。故在此選擇以智慧卡當做私密金鑰的儲存媒體，既不容易被複製，又具有標準化的方法可資應用。使用智慧卡的優缺點請見本文 2.3 一節。 3. 不過，智慧卡當做私密金鑰的載體時，智慧卡仍可能被偷走或是被合法使用者提供給非法電腦使用，因此必須讓此智慧卡僅能在特定的電腦上使用。所以智慧卡內除了私密金鑰外，尚會儲存電腦資訊，這是保證元件在特定電腦上的重要防護。
流程說明：	1. 「製作元件授權書」使用個案將元件資訊與電腦資訊傳送過來。 2. 依據電腦資訊向網站伺服器取得該電腦的公開金鑰以及伺服器本身的私密金鑰。 3. 使用流程 2 的兩把金鑰依序為原始的授權書加密。 4. 將加密完成的元件授權書回傳給「製作元件授權書」使用個案。

表 5：使用個案說明 – 製作受保護元件

名稱：	製作受保護元件
啟動者：	元件開發人員

限制條件：	「使用控管代理人」介面已經開發完成
其他說明：	1. 由於「受保護元件」在實際執行時會引用到「使用控管代理人」，因此「使用控管代理人」必須在每一台使用到「受保護元件」的電腦上執行。
流程說明：	1. 「元件開發人員」在「受保護元件」內引用「使用控管代理人」介面。 2. 依據樣板設定元件資訊。 3. 在元件啟動程序的最前面加入委派程式碼。 4. 編譯元件並上傳至網站伺服器集中管理。此時完成的元件即使用個案的其中一個參與者「受保護的元件」。

表 6：使用個案說明 – 請求「使用控管代理人」進行使用權檢查

名稱：	請求「使用控管代理人」進行使用權檢查
啟動者：	受保護元件
流程說明：	1. 當應用程式要求受保護元件的任一物件時，就會啟動此使用個案。 2. 受保護元件將元件資訊傳遞給「使用控管代理人」，整個驗證的工作便委派到「使用控管代理人」。 3. 如果驗證正確則直接結束本使用個案，讓元件得以繼續執行其他功能。若驗證失敗則會產生錯誤讓應用程式補捉。

表 7：使用個案說明 – 「使用控管代理人」控管使用權

名稱：	「使用控管代理人」控管使用權
啟動者：	請求「使用控管代理人」進行使用權檢查
其他說明：	與請求「使用控管代理人」進行使用權檢查使用個案合併的完整活動圖請見圖 11。
流程說明：	1. 當「應用程式」啟動「受保護元件」內的任一物件時，在該元件的初始化程序中會將使用控管檢查程序的工作委派給「使用控管代理人」，並傳送元件資訊讓代理人識別目前要求驗證的元件。 2. 「使用控管代理人」接收到驗證的請求時，會先檢查目前此元件在本次機器開機後，是否已經驗證過，如果驗證過則會有驗證結果，就會直接跳到步驟 6。若元件尚未經過驗證，則會繼續步驟 3。

	3. 代理人動態收集電腦資訊，並讀取智慧卡中的電腦資訊，比對以確定智慧卡是否在限定的電腦上使用。 4. 代理人再讀取智慧卡中的私密金鑰，並從元件資訊找出相對應的元件授權書。 5. 代理人以電腦的私密金鑰與伺服器的公開金鑰(已編譯在代理人程式內部)解開授權書，並比對所記載的資訊是否一致。比對結果記錄在元件授權狀態中。 6. 最後檢查元件授權狀態，若是授權正確則將程式執行權還給元件，繼續執行其他元件的功能；若是授權不正確或是以上任一步驟的比對有誤，則會停止繼續執行該元件，應用程式應補捉此可能錯誤提示使用者。

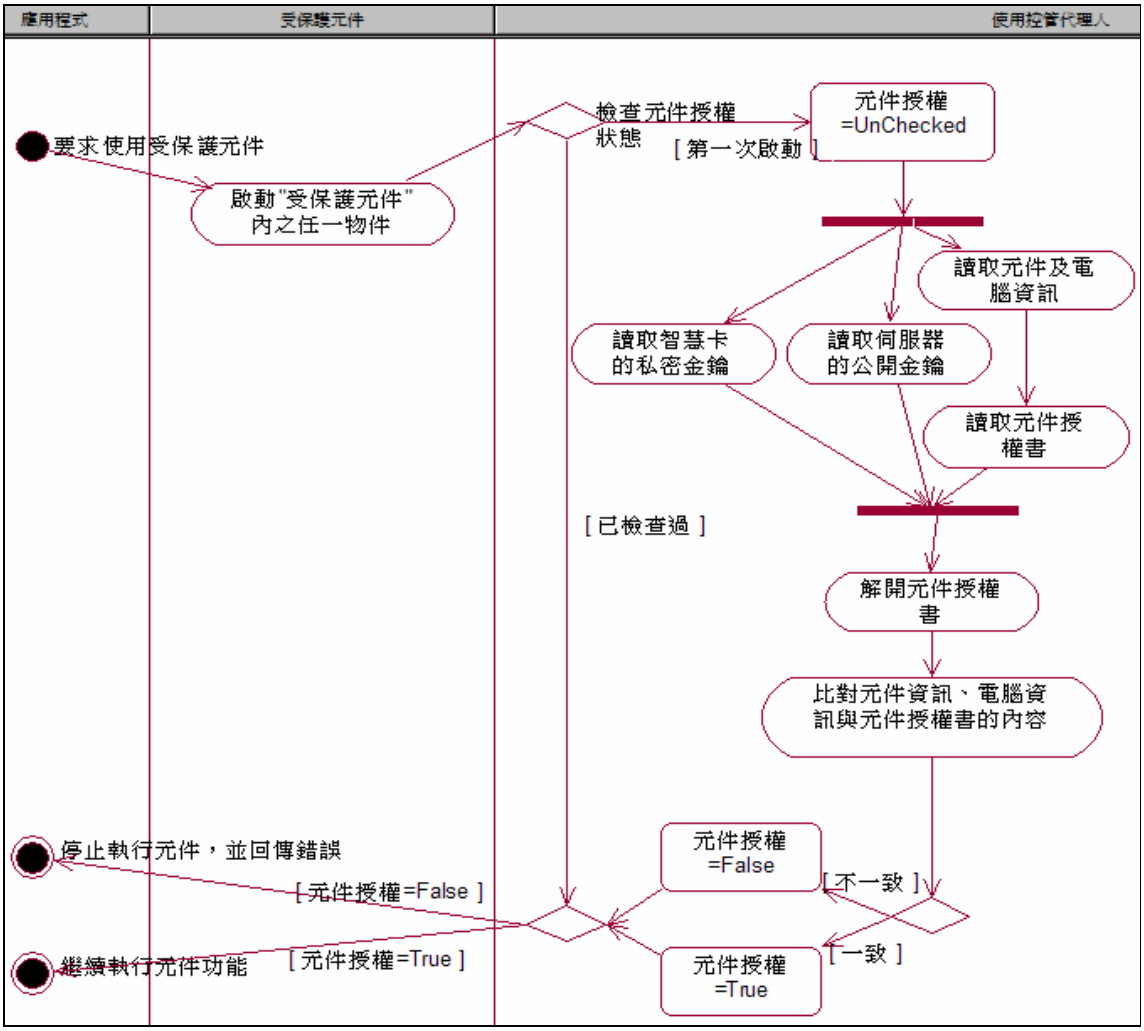

圖 11：智光科技-分析設計 - 「使用控管代理人」控管使用權活動圖

4.3.4 使用個案的類別圖與循序圖

以上的系統功能使用個案實現了系統需求的定義，每一個使用個案需要物件的互動以完成這些功能，因此以下需要針對每一個使用個案分析所使用的類別(以 UML 類別圖展示)，而類別圖是靜態結構，只要再以 UML 循序圖來表示類別的動態執行關係，就能夠提供實作時足夠的資訊，便可以進到下一個建構階段。類別圖的設計亦有階段之分，剛開始的階段可能只包括類別與關係，愈後期的階段愈接近實作的觀點，一個真實世界中的類別可能會因實作需求而分成數個類別，此時介面、方法、屬性這一些也會明確表示出來。這裡列示的每一類別圖並不複雜，所以直接以本階段的最後成果來表示，實際上每一個圖形都是經過多次反覆不斷修正得來的。

製作類別圖時最重要的一件事就是尋找類別，這項工作可以從已完成的使用個案描述來進行。一般來說，在使用個案描述中的名詞都可以是系統中的類別，而動詞即是這些類別所應該具備的方法。類別除了真實世界的人員、系統或設備等，不屬於本系統所需要提供的功能之外，剩下的類別很可能就是我們所需要的，然後再分析這些類別之間的關係，即可以繪出類別圖。

循序圖則是描述這些類別如何互動，因此只要從使用個案的流程中就可以找出這些類別的互動關係以及互動的順序。

因為這些使用個案是高度相關且功能並不複雜，因此其類別圖可以合併一起呈現，僅將上述的五個使用個案分成二個部分來製作類別圖與循序圖。

1. 製作元件授權書、為授權書加入防複製功能：

這部分的功能於之前的使用個案中已說明將實作成網站形式，因此這些類別將實作在同一個元件中，名稱為：GenLicense.dll，會在網站伺服器上執行。網站伺服器利用 ASP 網頁與元件開發人員互動以產生授權書。

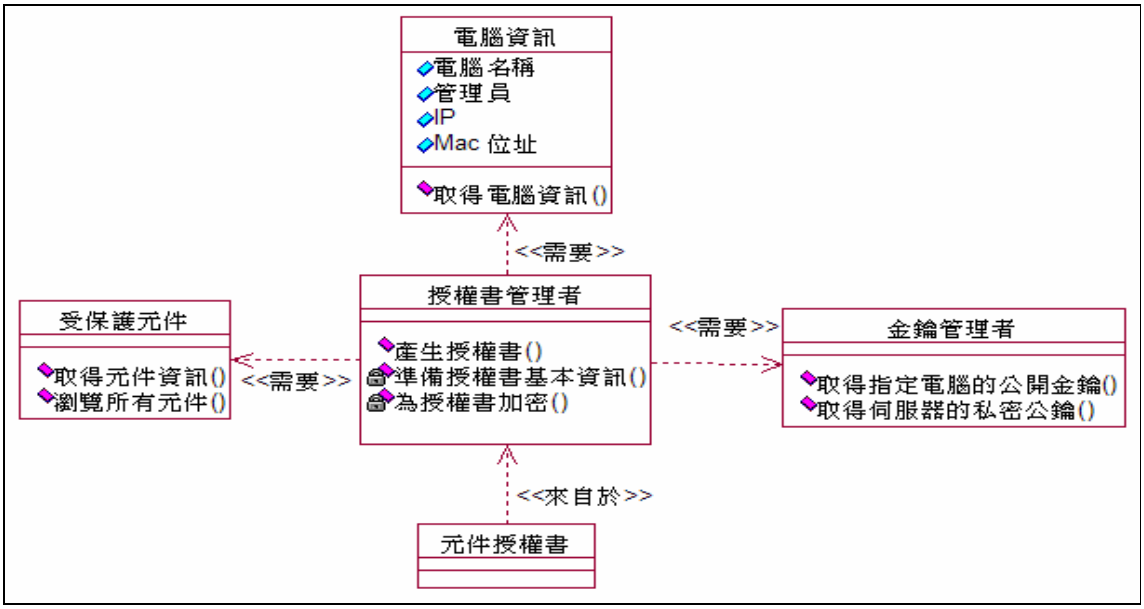

圖 12：智光科技-分析設計-使用個案類別圖 1

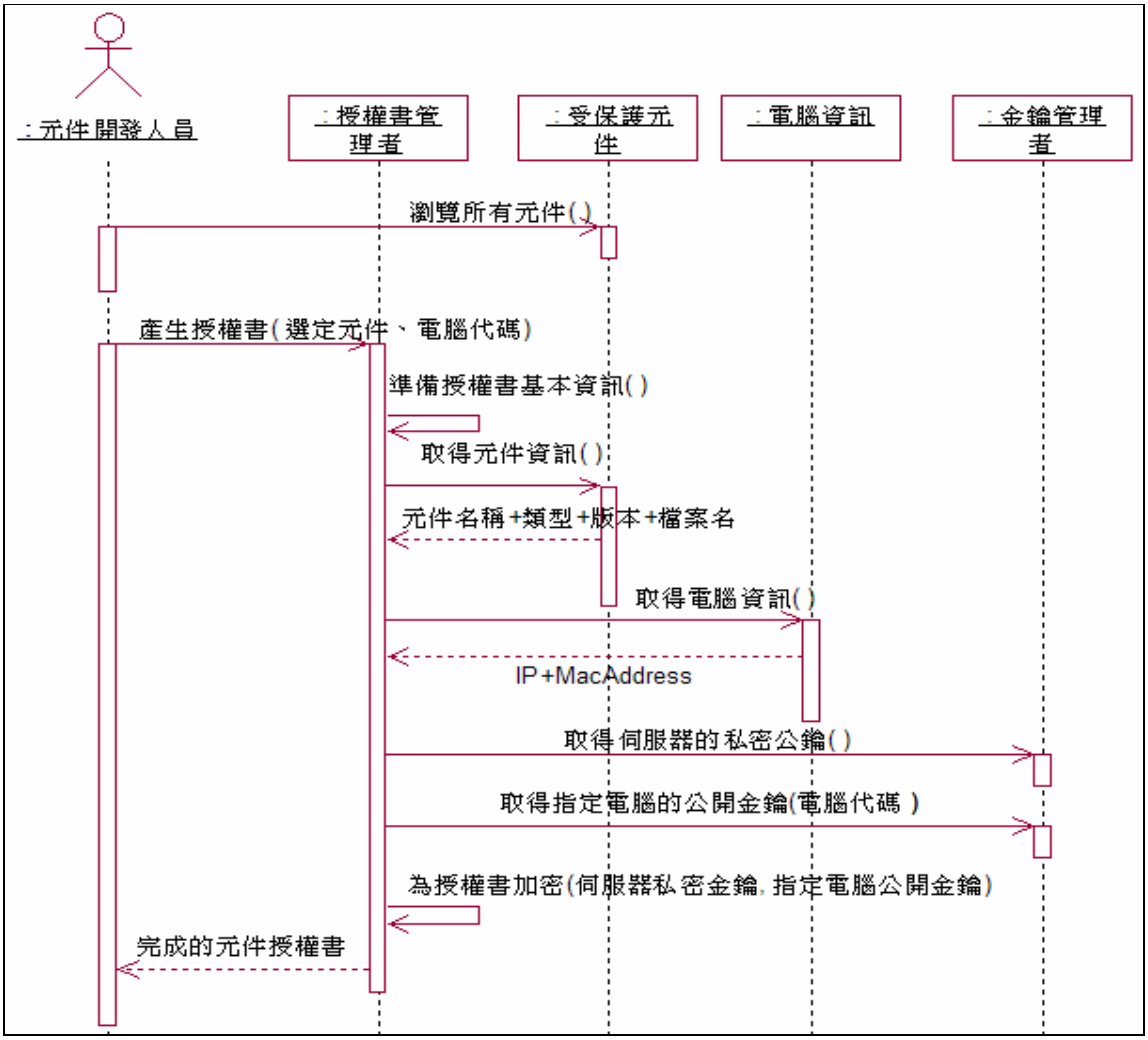

圖 13：智光科技-分析設計-使用個案循序圖 1

2. 製作受保護元件、請求「使用控管代理人」進行使用權檢查、「使用控管代理人」控管使用權：

這部分的功能如之前的使用個案所述，必須實作後與受保護元件一起遞送到使用的電腦上。其中的使用控管代理人與本機資訊兩類別合併成為一個元件：Agent.dll，而智慧卡管理者因為使用不同智慧卡可能會有不同的實作內容，為提供未來升級時的彈性，因此將它與代理人分開，成為另一個元件：CardReader.dll。

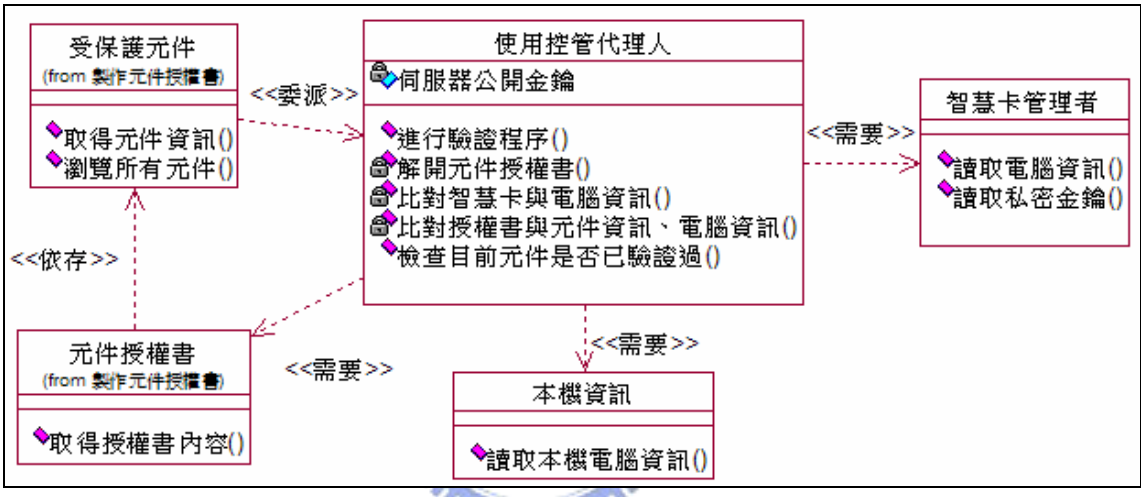

圖 14：智光科技-分析設計-使用個案類別圖 2

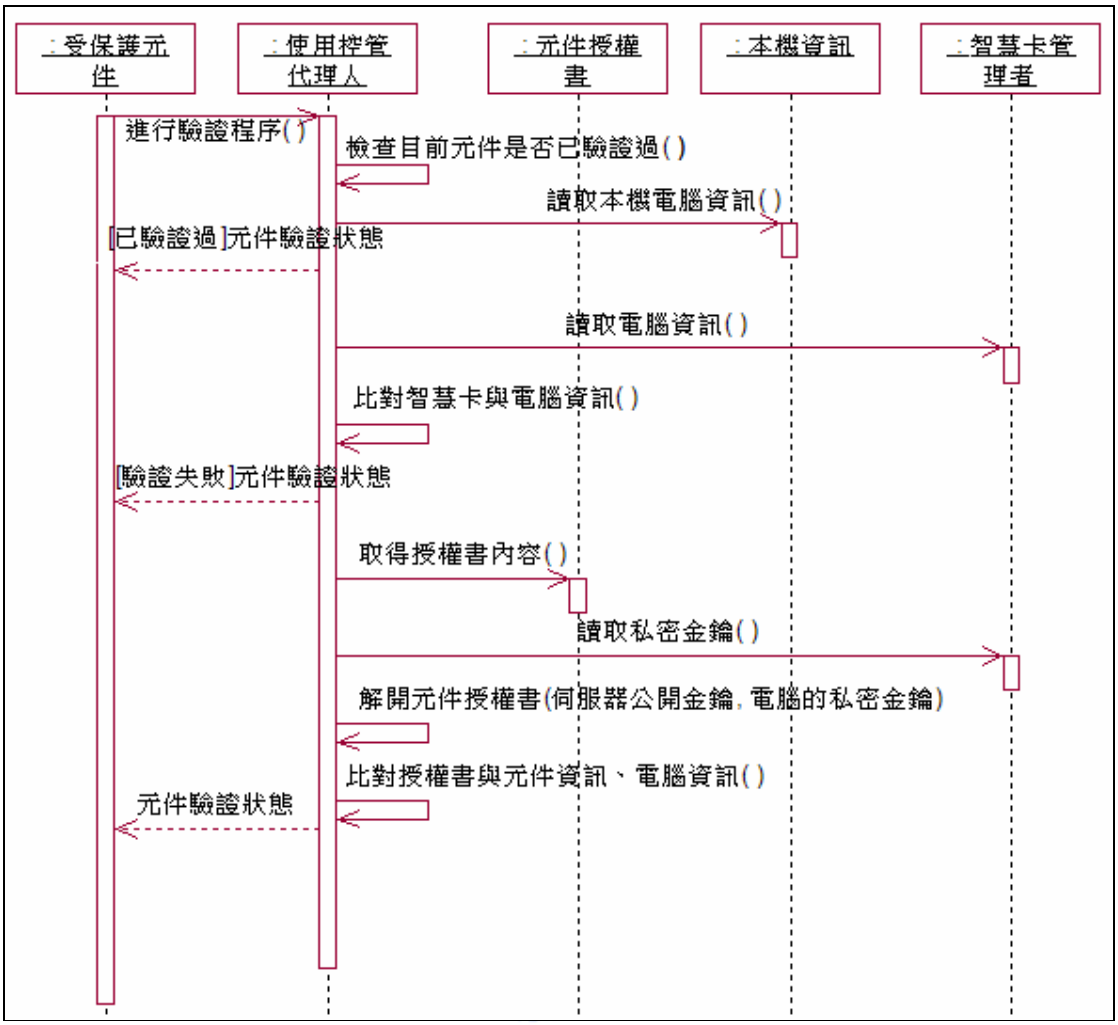

圖 15：智光科技-分析設計-使用個案循序圖 2

4.3.5 元件保護的佈署架構與元件圖

由以上的分析與設計中可以知道，本系統會使用到一個網站伺服器以及多個使用受保護元件的電腦，但是後者這些電腦並不會與網站伺服器在執行時有任何的溝通，需要遞送到使用受保護元件電腦的元件與授權書等可利用實體的光碟或是其他網站來完成，如果選擇以網站方式發送元件與授權書，其實是可以利用一個網站伺服器來完成，但是這是選擇性的設計，需考量的因素不在本研究的討論範圍，因此在佈署圖內並未將這兩者畫上關聯。

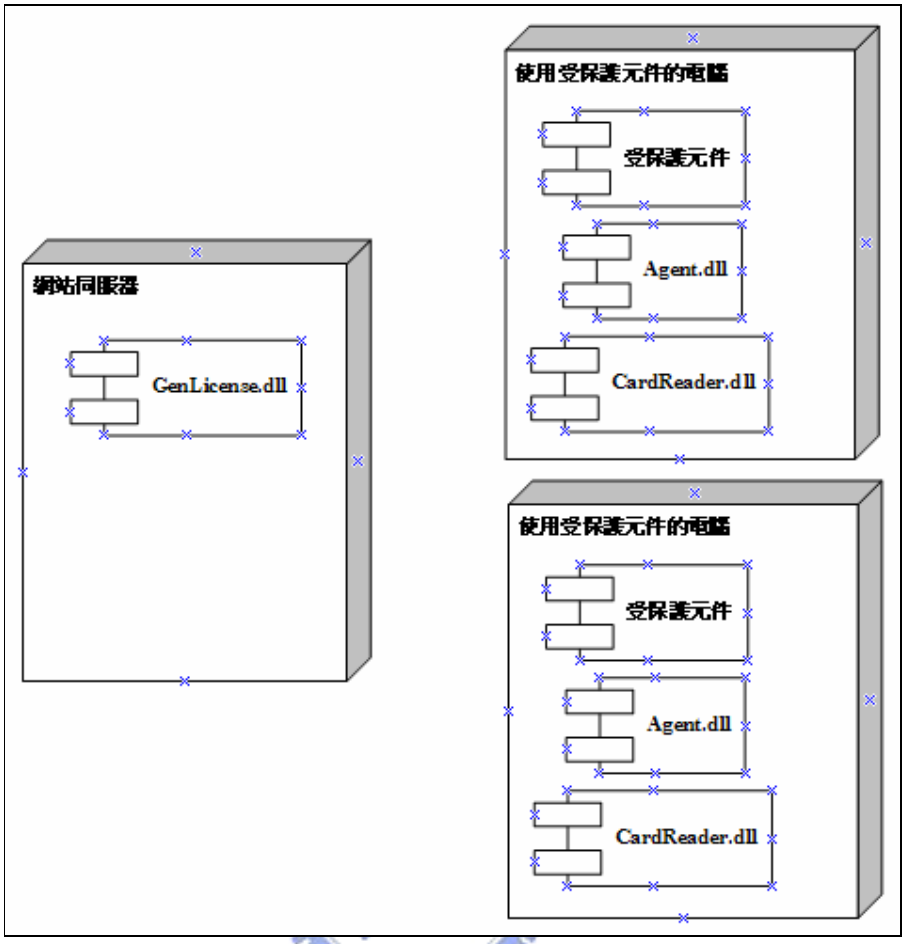

圖 16：智光科技-分析設計-元件佈署圖

至此已完成整個詳述階段的所有工作，下一階段便是將以上的設計實作出程式碼。

4.4 建構階段：實作

本階段實作的成果就是詳述階段最後的元件執行檔：GenLicense.dll、Agent.dll、CardReader.dll，以及製作元件授權書的網站伺服器網頁。網站的執行畫面可在最後的測試流程中看見，在此便不贅述。實作中需要特別說明的就是為受保護元件加入使用控管程式碼的部分，以下舉一個元件範例來說明：

開發部剛完成一個最新的壓縮與解壓縮元件，這個元件未加入使用控管程式碼之前的狀態如下圖所示：(元件的演算法細節已省略)

```
Option Explicit

Private Sub Class_Initialize()
    '元件的初始化動作
End Sub


Public Function Compress(bitstring As Byte) As String
    Compress = "已壓縮完成的資料"
End Function

Public Function DeCompress(bitstring As Byte) As String
    DeCompress = "已解壓縮完成的資料"
End Function
```

圖 17：受保護元件未加入使用控管程序

而已經加入使用控管程序的元件程式如下圖所示，比較後可以發現加入的程式碼都非常類似，因此可以做成樣版供開發人員複製使用，需要設定的部分已用橢圓形標示強調。

```
Option Explicit

Private Sub AutoCheckLicense()
    '每一元件需設定的資訊，同一元件的設定必須都一致
    Const comName As String = "Compressor"
    Const comType As String = ".Net"
    Const fileName As String = "Compressor.dll"
    Const comVersion As String = "1.1"

    Dim objAgent As New CAgent

    '將元件資訊傳送給驗證代理人
    objAgent.SetComInfo comName & "_" & comType & "_" & fileName & "_" & comVersion
    '將驗證工作委派給代理人，若無錯誤元件會繼續執行，若有錯則會在代理人中產生錯誤
    objAgent.AutoCheckLicense
End Sub


Private Sub Class_Initialize()
    '請求驗證，放在物件初始化的最前面
    AutoCheckLicense

    '元件的初始化動作
End Sub


Public Function Compress(bitstring As Byte) As String
    Compress = "已壓縮完成的資料"
End Function

Public Function DeCompress(bitstring As Byte) As String
    DeCompress = "已解壓縮完成的資料"
End Function
```

圖 18：受保護元件已加入使用控管程序

4.5 轉換階段：測試

上一個建構階段完成後，整個系統的功能算是均開發完成。最後這個階段要確認開發好的系統是否能達成原先的需求，並且將系統佈署完成以能順利運作。由於元件內部的測試與整合，其實在建構的實作流程中就會一併完成，因此在這個階段的測試工作，會將使用者的角度來測試其功能，因此以智光科技的元件開發與測試流程，設計使用受保護元件的情境，再實際看本系統在這些情境中是否完成預期的功能。

4.5.1 設計測試用模擬情境

從 3.1 節的智光公司案例說明中可以知道，該公司使用本系統就是為了使未上市的元件只能在測試部的電腦中使用，因此我們設計四個模擬使用情境，以展現本系統的功能。我們假設本系統的程式佈署已完成，測試部的每一台電腦所需要的金鑰對與智慧卡都已準備妥當。

1. 製作元件授權書：開發部剛完成一個高效能的壓縮/解壓縮元件，而且已經依照本系統提供的方式為該元件加入使用控管功能，準備將此元件轉交給測試部進行測試。遵循本系統的設計流程，本元件的使用必需搭配元件授權書一併使用，因此開發人員到網站伺服器上，瀏覽到本元件，然後指定測試部的其中一個電腦為目標電腦，這個電腦的 IP 是 192.168.234.2，MacAddress 是 AA-50-56-C0-00-08。這個情境若順利完成，預期會產生一個元件授權書的檔案。
2. 元件執行時沒有授權書：如果元件沒有搭配授權書就執行，照本系統的設計是無法正常使用的，因此預期會產生錯誤而停止元件執行。
3. 在正確的目標電腦上執行：如果元件搭配合法的授權書在正確的目標電腦上執行時，預期可以順利通過驗證程序，並且繼續執行所要求的元件功能。
4. 元件與授權書一併被複製到目標電腦以外的電腦上執行：雖然元件搭配著授權書執行，但由於授權書記錄的目標電腦資訊與正在執行的電腦資訊不符合，因此預期會產生錯誤而停止元件執行。

4.5.2 實作測試用的受保護元件

為了完成以上的模擬情境測試，我們實作該壓縮/解壓縮的元件：Compressor.dll，並且已加入使用控管功能。該元件提供二個函數，即 Compress 與 DeCompress，由於是

測試用的受保護元件，因此只會回傳一個固定字串，其元件程式碼如圖 18 所示。

4.5.3 實作測試用的應用程式

由於元件並沒有提供使用者介面，因此若要確認元件功能執行是否正常，必須另行實作一個應用程式來引用來元件。在此設計的應用程式非常簡單，只有一個輸入字串的文字框以輸入要壓縮的文字，然後有一個按鈕啟動 Compressor.dll 元件的壓縮功能。其應用程式的畫面如下圖所示：

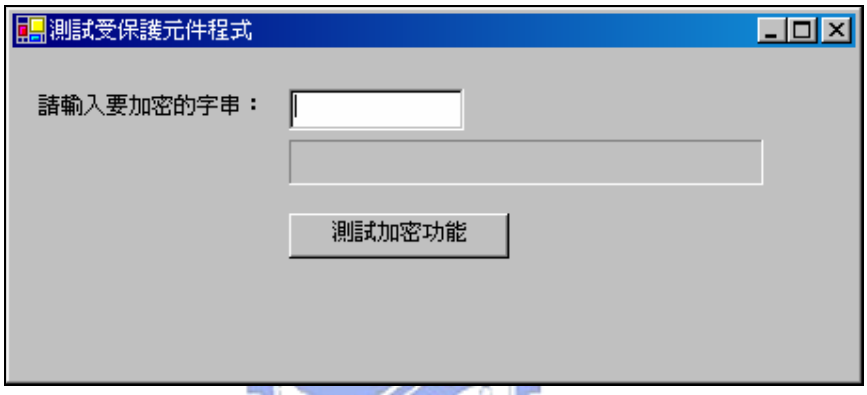

圖 19：智光科技-測試-測試用應用程式執行畫面

4.5.4 實際測試與結果

當以上的受保護元件與應用程式均準備妥當之後，接下來開始進行實際的測試。以下將依上述的四個情境進行，並將其執行畫面顯示於後。

1. 製作元件授權書：執行結果如預期，會產生授權書供申請者儲存。

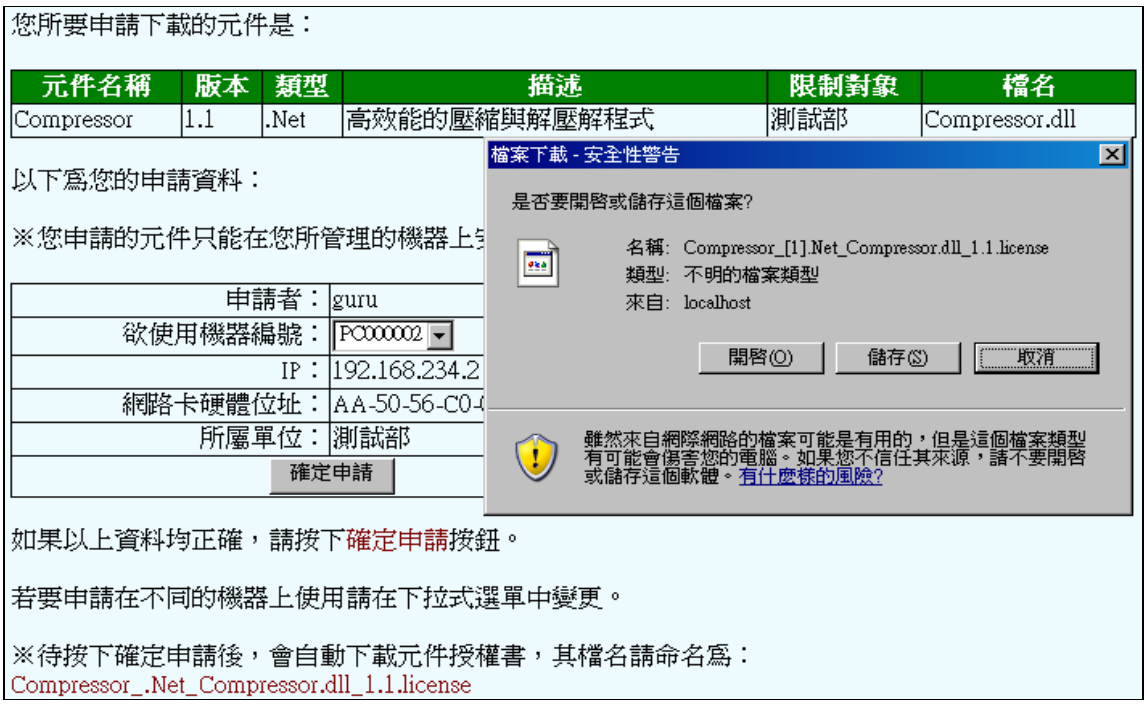

圖 20：智光科技-測試-製作元件授權書

2. 元件執行時沒有授權書：執行結果如預期，會產生錯誤訊息。

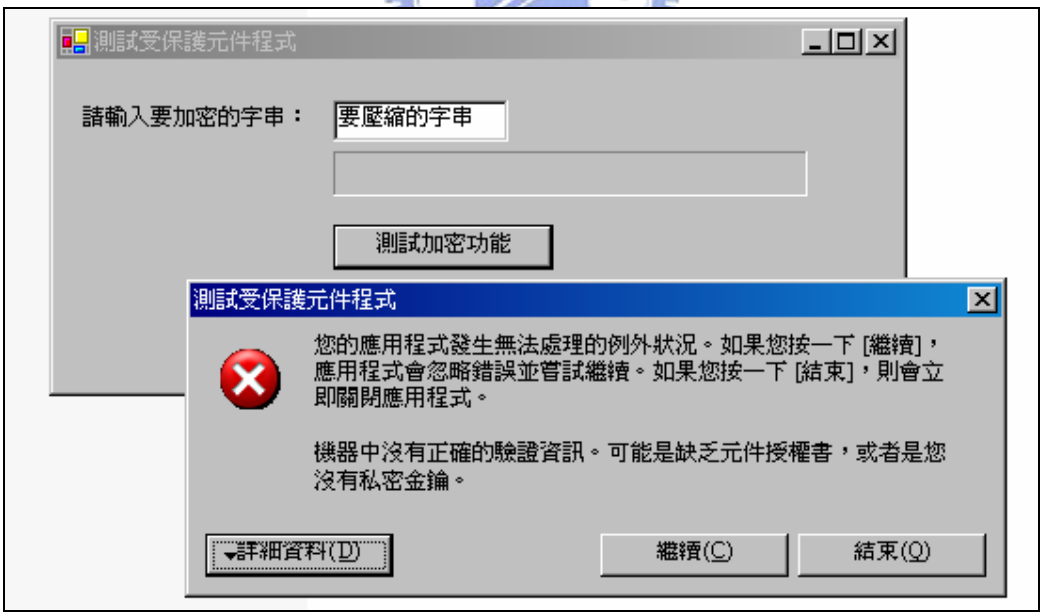

圖 21：智光科技-測試-沒有元件授權書

3. 在正確的目標電腦上執行：執行結果如預期，元件直接執行所要求的功能，並回應該功能的回傳值。

圖 22：智光科技-測試-正確的目標電腦上執行

4. 元件與授權書被複製到非目標電腦上執行：執行結果如預期，產生錯誤訊息。

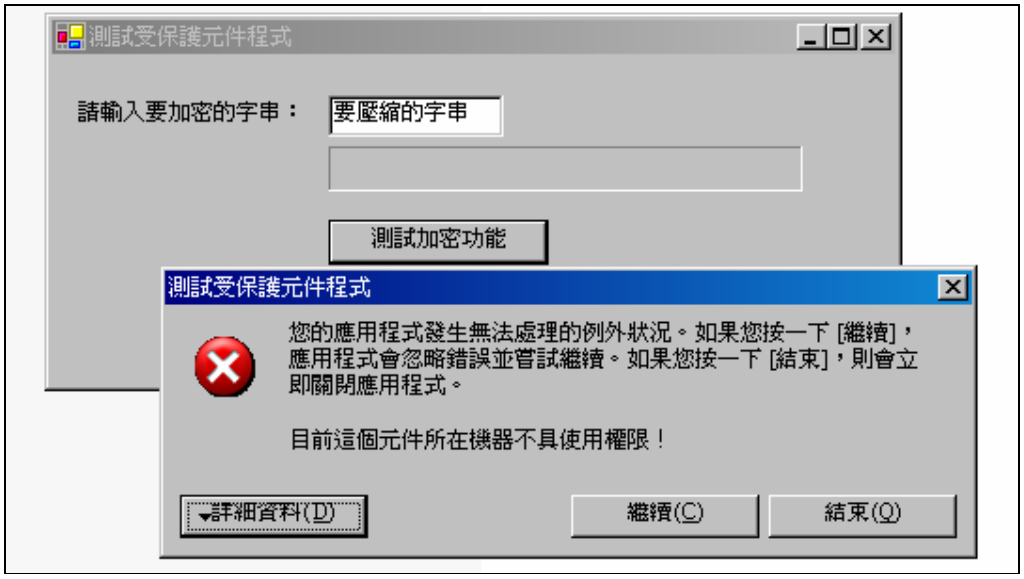

圖 23：智光科技-測試-在非目標電腦上執行

4.6 討論

本系統至此已完成整個系統的生命週期，在這個發展過程中，本文利用每個階段的產出討論了元件保護方法的需求、解決方案、系統設計等議題，最後開發出適用於智光科技軟體公司的元件保護方法。雖然在本專案中將此方法稱之為系統，但這個系統並非如一般資訊系統純粹由軟體構成，而是包括配套的程序、規範與支援性的程式。

這樣的系統雖然是以虛擬的軟體公司為對象所設計出來的，但是對於不同的組織，

只要有相同的需求，都可以採用本方法的架構，依該組織的特性或特別的需求略為修改就能使用。而且本系統能修改以結合組織內部門管理、人員管理、電腦管理等系統，讓整個製作授權書、下載元件、開發元件的過程更流暢而完整。因此或可稱本文的方法為一個參考性的架構。

從執行使用的層面來看，由於驗證程序是多出的執行步驟，必然會影響到整個應用程式的執行效率。其中用到非對稱加密法來解密授權書，雖然會減低一些效率，但本設計中的授權書內容僅包含元件資訊與電腦資訊，對於加解密運算量並不大，因此影響有限。本文由於著重在整體架構的建立，因此未能針對這些效能進行細部分析。但相信未來硬體設備的進步，這些問題的影響應該會愈來愈小。

雖然本研究試圖為元件授權提供一套完整又有效率的方案，但是在以下的情況仍然無法提供保護：

1. 元件、授權書、智慧卡、機器同時被帶走。
2. 驗證程序可能被某些駭客中斷跳過。

針對第一個問題，如果連同機器都被帶走的情況，非法使用者還是僅能在該台機器上執行一份程式，對於商業軟體來說，只要能避免被大量複製就算是可以接受的保護方式，因此尚不算是非常大的問題。對組織而言，這算是嚴重的竊盜行為，通常組織會有其他方法保護組織資產，所以也較不易發生。

而第二個問題目前來說的確很容易破解，但是程式碼本身的驗證與完整性，都有商業軟體及其他研究在進行。而本文的著眼點仍是整體程序與其帶來的效益，且這個問題是所有的程式都會有的共同問題，為避免整個研究失焦，並未探討這部分的防護方式。

五、結論與未來工作

5.1 結論

本研究的主要目的是為元件提供保護，讓元件只能在特定電腦上執行。而目前的研究中都聚焦在軟體保護的方法上，因此本文從軟體保護的方法開始探討，進而分析這些方法應用在元件環境下的適用狀況，利用物件導向系統開發的方法，開發出一套軟體元件的保護方法，以為元件加上使用控管的能力。

目前應用在軟體保護的方法的主要缺點是需要依賴網路與伺服器驗證、依賴硬體做為授權證明難以擴充到其他軟體、為加入授權驗證能力使得程式開發困難。本文針對以上的這些議題進行改善，主要的貢獻整理如下：

1. 限制元件僅能在已申請的電腦上執行。
2. 元件不需集中在伺服器端執行。
3. 使用控管的檢查過程不需仰賴網路，避免網路斷線時影響元件執行。
4. 以檔案做為授權資訊載體，以方便新增、更新授權。
5. 將共同的檢查程序抽出成代理人以便利元件加入保護能力。
6. 使用網站製作元件授權書，容易管理授權設定及變更授權。

雖然本文的研究對象是元件，但如果將此架構應用在一般性軟體的啟始元件上，就能使整個軟體受到保護。

5.2 未來工作

本文是初次將這種軟體保護的架構應用在元件上，因此將一些現實環境的需求簡化以突顯架構中重要的部分。在私密金鑰的儲存上採用智慧卡，除了安全性的考量之外，尚有一卡多用的功能，在組織管理上，許多公司已採用智慧卡做為門禁系統與電腦登入的身分識別，所以智慧卡內常存有身分的憑證。本方法中的智慧卡是以電腦對識別對象，但如果能結合組織或政府發行的自然人憑證使用，或許還能提昇軟體的安全性保護與便利性，也讓一般性軟體或元件的使用權可以適用於組織的分層授權與管理。只是元

件的使用者是應用程式，如果要將元件的限制對象設定為自然人，則尚有許多議題可以討論。

本架構所未能解決的另一個問題即是硬體失竊。由於本架構的設計是希望元件執行時能提供自我驗證的機制，不需依賴伺服器控管，而元件的限制對象是設定在機器而不是使用者，因此當整部機器失竊時，元件、授權書、智慧卡都一併被帶走時，就無法達到防範的效果。未來這部分或許可以將「使用者」加入限制對象，只是當限制愈多其執行時的互動可能就更為繁複，而且與元件的使用對象(是應用程式而非自然人)有關，在目前一機多使用者的情況恐怕會造成使用時額外的負擔，或許未來的研究能解決這方面的問題。

六、參考文獻

1. 吳仁和，《物件導向系統分析與設計：結合 MDA 與 UML》，智勝文化，台北，2005。
2. 林祝興，李鎮宇，「網路軟體保護方法之研究：一次安裝方案」，二〇〇〇網際網路與分散式系統研討會論文集 I，438-441 頁，台南，2000。
3. 林清展，「軟體使用權控管機制之研究」，靜宜大學，碩士論文，2001。
4. 查修傑，連麗真，陳雪美譯，「電子商務概論」，和碩科技，台北，1999。
5. 梁伶君，「智慧卡簡介與應用趨勢」，成功大學圖書館館刊，第四期，1999/10，取自：http://www.lib.ncku.edu.tw/journal/journal/4/6_1.htm。
6. Tuomas Aura, Dieter Gollmann, “Software License Management With Smart Cards”, *USENIX Workshop on Smartcard Technology*, pp. 75-85,Chicago, May 1999.
7. Bernd Bruegge and Allen H. Dutoit, “Object-Oriented Software Engineering : Using UML, Patterns, and Java”, 2nd ed.(international), Pearson Prentice Hall, 2004.
8. N. S. Gill, P. S. Grover, “Component-based measurement : few useful guidelines”, *ACM SIGSOFT Software Engineering Notes*, Volume 28 Issue 6, November 2003.
9. Patrick C.K. Hung, Kamalakar Karlapalem, “Security and Privacy Aspects of SmartFlow Internet Payment System,” *Proceedings of the 32nd Hawaii International Conference on System Sciences*, 1999.
10. Philippe Kruchten, 《The Rational Unified Process, An Introduction》, 2nd Ed., Addison Wesley, 2000.
11. Tomas Sander, Christian F. Tschudin, “On Software Protection Via Function Hiding”, *2nd International Workshop on Information Hiding*, 1998.
12. Shiuh-Pyng Shieh, Chern-Tang Lin, Shianyow Wu, “Optimal Assignment of Mobile Agents for Software Authorization and Protection”, *Computer Communication*, 22, pp. 46-55, 1999.