私立東海大學資訊工程與科學研究所

碩士論文

指導教授：周忠信

數位創新方法論的探討

The study of methodology for digital innovation

研究生：陳丁加

中 華 民 國 九 十 九 年 十 一 月 二 十 三 日

摘要

隨著數位技術越來越蓬勃發展，產品數位化的範圍也越來越廣。台灣身為數位產品的主要製造商，運用數位創新(digital creativity 或 digital innovation)於其他傳統領域產品，對於台灣產業的未來發展乃極為重要。所謂數位創新是指運用數位技術於新產品或新應用的發想、發掘、與發明上。過去有關創新研究的最主要成果，則以 TRIZ 最具代表性。但是隨著數位技術的發展，TRIZ 的四十條發明原理已有點過時，無法完全應用到數位創新上。因此本論文基於 TRIZ 與數位技術創新常常面對的問題，另行提出一個新的創新方法論。此方法論以設計樣式(design patterns)為精神，描述三十六個發明原理。這些發明原理除了可以涵蓋原來 TRIZ 的四十條與數位創新有關的發明原理外，我們還增加了 TRIZ 未考慮到的新原理。此外，此三十六條發明原理各以成語命名，有助於發明過程時的運用。本論文最後以 RAID 為案例，說明如何運用此三十六條發明原理。

關鍵字：數位創新、TRIZ、發明原理、RAID

Abstract

With rapid development of digital technology, more and more products with digital flavors are invented. Taiwan, as one of the major super strength in digital world, utilizing digital technology to traditionally non-digital products and service is extremely important. Adopting digital technology in the new product or service design is called digital innovation. Methodology with invention principles is critical when inventing new solutions for digital innovation. The most popular invention methodology available nowadays is called TRIZ. However, TRIZ's 40 invention principles are not completely applied to digital innovation. Therefore, in this thesis we propose a new set of invention principles for digital innovation. The 36 new principles can cover the original TRIZ's 40 principles and in fact two of them are whole new ones. In order to make the principles to be more easily used, we further follow the syntax of design pattern style to define the invention principles. At the end of the thesis, example scenarios based on RAID are used to explain how the 36 new principles are applied.

Keyword: digital innovation, TRIZ, invention principle, RAID

目錄

表目錄

圖目錄

第二章

第四章

第一章 研究動機與目的

1.1 研究動機與目的

專家在專長領域面臨到的問題，可以透過經驗與知識推演，進而解決問題。如果專家企圖解決專長領域以外的問題，當過往的經驗不足以解決問題時，就需要經由創新來解決問題[7][35]。美國紐約水牛城州立大學的「國際創造力研究中心」詳列了 12 項創造力研究的重要理由，分別是(1)發展越過智慧極限的潛能、(2)工商業的快速發展、(3)有效地應用組織內的人力資源、(4)發展更新更好的問題解決方法、(5)社會的發展、(6)建立固有的知識、(7)人先天上的能力、(8)精神健康的論點、(9)日益成長的關注、(10)所有學科的基礎、(11)有助於有效率的領導和(12)強化學習的過程 [7]。由上述可知，創新可以幫助解決許多日常生活中會遇到的問題並幫助自我提昇，進而加速產業發展和提高國家競爭力。

隨著數位技術越來越蓬勃發展，數位創新(digital creativity 或 digital innovation)也越來越受到重視[5][13][15][31]。所謂數位創新是指利用數位技術於發想、發掘、與發明新應用或新產品。數位創新應用的領域非常廣，舉凡電子產品、傳統商品、文化產業、設計產業、建築產業、網路產業、企業營運、政府治理等，都可利用數位技術來創新、發明。透過數位技術，我們可以解決許多以往無法解決的問題；透過數位技術，我們也可以發想出以往無法做到的東西與服務。

然而，創新是一件很困難的事。當我們面臨一個問題時，容易在找尋解決方法的過程中，因為思考的廣度與深度不足，造成無法解決問題。這是因為要解決某個領域的問題，通常參與討論的對象，都是同領域的專家，有時候會因為當局者迷，而誤判了情勢。TRIZ 的出現，某種程度可以解決此類問題[2][3][4][9][16][26][27][29][33]。TRIZ 近年來在創新領域受到許多討論，有很多應用都相繼出現[4][8][12][17][18][19][24][25][34]。TRIZ 是由俄國科學家 Genrich Altshuller 與他帶領的團隊所發展，是一個創新解決問題的方法論。TRIZ 能夠有系統的帶領使用者，一步一步地解決問題。TRIZ 的理論基礎是認

為，這世界上大部份創新解決問題的方法，都可以被歸類為某種模式。而同樣的模式可以用來解決不同領域但是相同類型的問題。舉例來說，當瓦斯不用時，如果不關掉，會有瓦斯外洩產生的安全問題，但往往會因為使用者一時忘記而造成憾事發生。所以創新發明就是自動偵測使用者是否有在用瓦斯。如果判斷不使用，則自動關掉。當開車時遇到需要緊急煞車的情況，這時需要疊煞，車子比較容易停住，但往往會因為駕駛人緊張，而緊踩煞車，這容易造成車輪鎖死，會有安全上的問題。所以創新發明就是自動偵測使用者緊踩煞車的情況，當有這種情況，則自動轉成疊煞模式，這也就是所謂的防鎖死煞車系統。以上這兩個案例屬於不同領域的問題，但卻都採取了相同的解決模式：當有問題產生時，透過自動偵測產生對應處理，進而防止使用者因不當行為而產生的問題。

Altshuller 與他帶領的團隊，自 1946 年開始發展 TRIZ，從 20 萬件專利當中選出了 4 萬件較具創新意義的專利，從中歸納分析。到了 1971 年 Altshuller 發表了 TRIZ 的衝突矩陣與 40 條發明原理。在那個年代，數位技術不發達，幾乎所有案例都與數位發明無關。雖然 TRIZ 的發明原理，可以解決跨領域問題，但是隨著電腦軟硬體技術的發明，創造了 Altshuller 時代不能想像的新解決方式。以往有些需要耗費很多資源才能解決的問題，現今或能運用數位技術而輕易解決。因此，雖然發明原理是抽象化的概念，使用者應該要延伸發明原理的定義，但是，在數位創新中，Altshuller 原有的一些發明原理確實很難套用數位技術來思考。

為解決上述問題，本論文主要目的是重新發展並整理數位創新的方法論。經過整理原來 Altshuller 的四十條發明原理，同時考慮數位技術的特性，我們重新定義出三十六條發明原理。此三十六條發明原理，主要用來解決數位創新過程中會遇到的問題。同時為方便學習與溝通，命名(naming)是協同合作與學習原理的關鍵之一。因此，本論文將此三十六條發明原理以設計樣式(design patterns)的精神撰寫。同時，每條發明原理的名稱，也以一個適當的成語作為命名。本論文最後以 RAID 產品作為範例，解釋如何運用此三十六條發明原理於解決 RAID 產品發明上的困難。

1.2 論文架構

本論文第二章將以 TRIZ 相關背景知識介紹為主。第三章則探討數位創新的三十六條發明原理。第四章涵蓋以 RAID 創新過程為例，說明如何運用數位創新方法論。第五章為本論文的結論與未來可能的研究方向。

第二章 TRIZ

TRIZ 是俄文字首的縮寫，代表"Theoria Resheneyva Isobretatelskehuh Zadach"這四個字，英文為"Theory for Solving Inventive Problems"，中文則翻做"創意問題的解決理論"。今日 TRIZ 是一個方法論、工具庫、知識庫和模式為基礎的科學，用來產生創新的思維與解決問題的方法。TRIZ 有系統地解決問題，有別於傳統上藉由隨機性的靈感方式來解決問題。

TRIZ 的發明創始人為 Genrich Altshuller，為前蘇聯海軍專利局的專利審核員。在專利審核過程中，Altshuller 發覺每一項專利的創新過程背後，都存在一定的模式，可以被歸類成為某種型態。Altshuller 認為發明只不過是在某些原則的幫助下解決技術上的矛盾，而這些矛盾則是所謂的需求衝突。這些解決矛盾的基本原則會被一再使用，或是會參考其他領域解決問題的基本原則。因此，如果發明家在面臨到相同或類似的矛盾時，能夠去參考其他發明家解決問題的基本原則，就可加快發明工作的進行。於是 Altshuller 與他所領導的團隊，從 1946 年開始便在 200,000 件專利中挑出 40,000 件較具創新意義的專利，整理這些專利解決問題的方法，進而歸納出這些方法的基本模式。

Altshuller 認為當創新的過程中遇到問題時，使用者想要改善一個工程參數，常常伴隨發生的是導致另外一個工程參數惡化。傳統的方法是用妥協的方式，而 TRIZ 則是用消除的方法，利用整理出來的基本模式，也就是發明原理，總共有 40 條發明原理提供給使用者參考。依照 Altshuller 團隊的分析歸納，經常遇到技術矛盾有 39 個，稱為 39 個工程參數，將這些工程參數對照表整理成矩陣的形式，就是所謂的 TRIZ 矛盾矩陣。矛盾矩陣為一 39×39 的矩陣，在其 1263 個欄位上提供發明原理。有些欄位提供一條以上，但最多不超過四條的發明原理。

接下來依序介紹 39 個工程參數、矛盾矩陣和 40 條發明原理。最後再介紹 TRIZ 如何使用 39 個工程參數、矛盾矩陣和 40 條發明原理解決創新過程中會遇到的問題，並利用一個實際案例來說明整個解決步驟。

2.1 TRIZ 的工程參數

表 2.1 為 TRIZ 的 39 個工程參數。使用者需要以抽象化的概念思考工程參數。舉例來說，要改善電腦記憶體的儲存容量，這 39 個工程參數中並沒有儲存容量這個名詞，所以必須靠使用者聯想。記憶體是固定物件，儲存容量是個虛擬概念，在實體上空間越大容量越大，空間大代表體積大，所以可以將記憶體的儲存容量轉化為固定物件的體積，而這正是編號第八的工程參數。

表 2.1 TRIZ 的 39 個工程參數

編號	工程參數	編號	工程參數	編號	工程參數
1	移動物件的重量	14	強度	27	可靠度
2	固定物件的重量	15	移動物件 的耐用性	28	測量精準度
3	移動物件的長度	16	固定物件 行動的耐用性	29	製造精準度
4	固定物件的長度	17	溫度	30	物件影響的 有害因子
5	移動物件的面積	18	亮度	31	物件生成的 有害因子
6	固定物件的面積	19	移動物件 消耗的能量	32	容易製造
7	移動物件的體積	20	固定物件 消耗的能量	33	容易操作
8	固定物件的體積	21	動能	34	容易維修
9	速度	22	能量損失	35	適用性與 多功能
10	力量	23	物質損失	36	裝置複雜性
11	張力、壓力	24	資訊遺失	37	偵測與量測 的困難度
12	形狀	25	時間浪費	38	自動化程度
13	目標物組合 的穩定性	26	物質的數量	39	生產力

2.2 矛盾矩陣

接下來介紹矛盾矩陣表，因為整個矛盾矩陣表很大所以用簡表的方式呈現。表 2.2 為矛盾矩陣簡表。縱軸為想要改善的工程參數，橫軸為會惡化的工程參數。矛盾矩陣是 Altshuller 團隊分析與歸納 4 萬件專利而成，使用者可以根

據想要改善的參數查詢這個矩陣，藉此獲得發明原理。所謂的發明原理是之前發明家解決問題的模式，使用者可以參考這些模式來解決問題。舉例來說，若使用者想要改善的工程參數是"移動物件的長度"，但相對會惡化的工程參數是"移動物件的重量"，則藉由矩陣告訴使用者，可以利用發明原理 8、15、29 和 34 去解決問題。不一定要把全部的發明原理都用上才可以解決問題，也可以只使用其中的某幾條發明原理。

表 2.2 矛盾矩陣簡表

惡化 / 改善		移動物件的重量	…	資訊遺失	…	生產力
		1	…	24	…	39
移動物件的重量	1		…	10, 24, 35	…	35, 3, 24, 37
固定物件的重量	2		…	10, 15, 35	…	1, 28, 15, 35
移動物件的長度	3	8, 15, 29, 34	…	1, 24	…	14, 4, 28, 29
…	…	…	…	…	…	…
生產力	39	35, 26, 24, 37	…	13, 15, 23	…	

2.3 發明原理

表 2.3 為發明原理名稱表，詳細的說明請參考文獻[28]。

表 2.3 發明原理名稱表

編號	發明原理	編號	發明原理	編號	發明原理	編號	發明原理
P1	分割	P11	預先緩衝	P21	跳過	P31	有孔材料
P2	抽離	P12	等位性	P22	利用壞物質	P32	改變顏色
P3	區域品質	P13	另一條路	P23	回饋	P33	同質性
P4	非對稱	P14	曲線	P24	中介物	P34	丟棄與回覆
P5	整合	P15	動態	P25	自我服務	P35	參數改變
P6	多工	P16	過多或過少的動作	P26	複製	P36	相轉變

P7	重疊娃娃	P17	另一個維度	P27	便宜短生命期的物件	P37	熱膨脹
P8	反重量	P18	機械震動	P28	機械取代	P38	強氧
P9	預先反作用	P19	週期動作	P29	氣動與液壓	P39	惰性氣體
P10	預先行動	P20	持續有用的動作	P30	彈性框架與薄膜	P40	複合材料

2.4 TRIZ 應用

接下來介紹 TRIZ 解決問題的基本流程。

圖 2.1 為 TRIZ 解決問題的流程示意圖。傳統上發明家解決問題，是在其經驗底下透過靈感與嘗試錯誤，來找到解決問題的方法。而 TRIZ 的方法是使用者將"問題"分析後，參考 39 個工程參數，設定要改善與會惡化的工程參數，這個步驟叫做"建立矛盾"。建立矛盾後，再去矛盾矩陣查表，查出相對應的"發明原理"。再參考發明原理所提供的概念，將概念具體化為"答案"。以上就是利用 TRIZ 找出解決問題的創新方法的基本流程。

舉例來說，有一個基礎網路建設廠商，其公司的產品有一個利用無線訊號傳輸資訊的設備，這個設備資訊傳輸的有效距離是一公里內。如果距離超過這個範圍，資訊容易受到雜訊干擾，會造成資料遺失。所以這家廠商想要改善這個問題，希望增加資料傳輸的距離，但又不希望資料遺失。我們可以使用 TRIZ 來思考這個問題。首先，將問題確認，"想要增加資料傳輸的距離，又不希望資料遺失"。再來是利用工程參數"建立矛盾"。資料傳輸的距離可以抽象化想成是移動物件的長度，這正是第 3 個工程參數，所以想要改善的工程參數是移動物件的長度。但一增加資料傳輸的距離，會使得資料遺失，這可以抽象化成資料遺失，這正是第 24 個工程參數。所以整個問題可以抽象想成是"要改善的工程參數是移動物件的長度，但相對應會惡化的工程參數是資料遺失"，而這就是建立矛盾的過程。建立矛盾後，就去矛盾矩陣查發明原理，得到兩條發明原理，分別是"P1-分割"與"P24-中介物"。找到發明原理後，我們就可以開始參考發明原理來發展我們的創意。最後是"答案"，而利用這兩條發明原理想出來的解決方案是利用中間傳輸設備當作中介轉換傳輸的據點，延長有效傳輸的距離。簡

單來說，如果資料傳輸的距離是 10 公里，但有效傳輸的距離是 1 公里，則將 10 公里，利用"分割"這個發明原理，將 10 公里拆成 10 等份，資料傳輸變成一公里接一公里傳輸，然後利用"中介物"這個發明原理，製作一個中介轉換傳輸的設備當作連接，讓資料傳輸就像是接力賽跑一樣。這樣一來，資料傳輸的距離就可以拉長，又可以防止資料遺失。

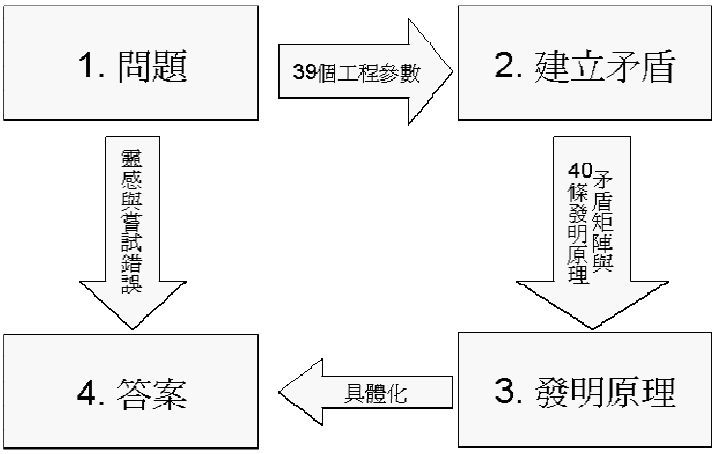

圖 2.1 為 TRIZ 解決問題與傳統上解決問題的流程示意圖

探討上述的案例，在 Altshuller 的時代不可能有這樣的專利，但我們經過 TRIZ 所想到的解決方案卻又正好可以解決我們的問題，這種為了增加距離，但一增加距離就會遺失資料的案例有很多，例如電報傳輸或電力傳輸，都是這類型的案例。這正符合 TRIZ 的基本理論，同種模式的解決方案常在不同領域中被使用。

第三章 數位創新發明原理

本論文在第二章中介紹 TRIZ 的作法與應用。然而在數位技術時代中，由於數位技術創造出許多新的可能性，是當時 Altshuller 建立 TRIZ 時所無法想像。舉例來說，一片 DVD 如果是單面單層，儲存容量是 4.7G。單面雙層，儲存容量是 8.5G。雙面雙層，儲存容量則可達 17G。容量越大，開發的難度越大。開發廠商想要增加 DVD 光碟的儲存容量，勢必會增加發明的困難度。如果利用 TRIZ，要改善 DVD 光碟的儲存容量，所以要改善的參數是”8-固定件體積要增加”，但相對會變壞的參數則是”36-裝置複雜性會變高”。藉由矛盾矩陣查得到的發明原理是，”P1-分割”以及”P31-有孔材料”。這兩條發明原理應用到 DVD 光碟上都無法使用，因為 DVD 光碟無論是實體或是抽象上都無法分割與打孔，所以這兩條發明原理都無法幫助找到解決問題的創新方法。這也證明了 TRIZ 要應用到數位創新上，勢必要經過修正。

本章我們將提出新的發明原理共三十六條。此三十六條發明原理，主要目的是將數位技術所衍伸的可能創新方法，融入到發明原理中。同時，此三十六條發明原理也可以完全對照 TRIZ 的 40 發明原理。有關與 TRIZ 的異同處，也將在本章的第二節中討論。

3.1 原理介紹

首先，我們將發明對象稱之為「主體」。主體則由單一或許多「組件」所構成。而主體與各組件間可透過各種可能「介面」結合，並經由對應「步驟」之「操作」，進而產生所需「行為」。

通常發明的主體，透過組件或行為來滿足發明的「需求目標」。需求目標一般則包括「功能性需求」與「非功能性需求」。功能性需求指該發明功能上所需達到的目標；非功能性需求則以該發明必需滿足的其他期望，如效率、速度、反應時間、資源耗損率等為主。為了達成設計目標，有時候除了必要組件外，尚需利用額外非必要的組件，來幫助達成需求目標。此類非必要組件稱之為「輔助體」。

發明的最大挑戰，乃在於必須解決發明主體、需求目標、使用「資源」以及其所在之「環境」，因交相互動而造成的各種「衝突」問題。

以下，各發明原理會分成「原理名稱」、「直觀含義」、「適用問題」、「運用說明」等四個部分，分別介紹各發明原理。

1. **原理名稱**：照貓畫虎

   直觀含義：利用已知的經驗，解決未知的問題。

   適用問題：

   在發明過程中，面臨到的難題是：為滿足需求目標，主體的設計對於發明者而言乃前所未見。

   運用說明：

   「照貓畫虎」發明原理的基本精神在於，首先尋找過去類似的需求目標以及其對應已知的設計，再根據需求、目標、資源、與環境等的差異與限制，從中挑選可作為模仿參考的對象，再行修改此設計或學習其設計，發展符合本次需求目標的新設計。

2. **原理名稱**：層次分明

   直觀含義：將一個整體分割成各自獨立的階層。

   適用問題：

   在發明過程中，面臨到的難題是：主體頗為複雜、很難處理。而且當部分組件需要變動時，卻又會牽一髮而動全身，造成只為修改局部，卻要變動全部。既不容易滿足需求目標、又會耗費許多資源。

   運用說明：

   「層次分明」發明原理的基本精神在於，根據整體內部彼此關係密切的組件，一塊塊分割出來。被分在同一塊內的組件間，必須具備極大的連動性；被分在不同塊間的組件，除連動關係必須越小越好外，各塊與塊間的介面關係，也必須能夠被明確規範與描述。當某塊被修改時，其他各塊可以不跟著也受影響。

3. **原理名稱**：抓大放小

直觀含義：

以「大者」為主，不需考慮、甚至忽略「小者」。

適用問題：

在發明過程中，面臨到的難題是：主體本身與其組件的需求目標，存在諸多的不一致性。若要全體兼顧，則可能造成資源浪費；或是受限於資源，造成無法滿足關鍵需求目標。

運用說明：

「抓大放小」發明原理的基本精神在於，找出會產生衝突的組件，轉移、降低、甚至去除其對於整體的重要性。因此，在解決問題上，就不需再顧慮此部分組件，從而得以在有限資源上達成整體的需求目標。

4. **原理名稱**：執兩用中

直觀含義：

不偏不倚，選擇中道而行。

適用問題：

在發明過程中，面臨到的難題是：當需求目標、環境、資源或組件之間存在魚與熊掌不可兼得、卻又必須並存的衝突時，不管選擇偏向哪一方，都可能會造成運作失衡、不易維護、彈性不足，進而最終無法達成真正目標。

運用說明：

「執兩用中」發明原理的基本精神在於，首先找出「魚」與「熊掌」兩方的關鍵介接關係。若此介接屬於雙方合作關係，為避免因某一方直接依附於另一方所成的高耦合度缺陷 – 也就是說，當一方改變時，即使另一方不准變也不行。此時，利用加入另一新組件做為兩方的中介者，乃可降低合作兩方的耦合關係；若此介接屬於資源競爭關係，此時可將資源採折衷方式分配。雖然只能滿足雙方部分所需，但卻因此而能維持整體運行。

5. **原理名稱**：化整為零

直觀含義：

一個完整的主體，由許多很小但相似的組件所構成。

適用問題：

在發明過程中，面臨到的難題是：主體本身包含一巨大組件，但由於該組件太大、或是複雜度太高，若要真正以單一組件設計之，則將耗費大量資源。

運用說明：

「化整為零」發明原理的基本精神在於，首先確認主體中，造成該組件巨大或高複雜的需求原因。再來則根據需求原因，尋找能夠滿足類似需求但卻遠為小型或價廉的組件。儘管個別低廉的類似組件無法滿足完整的需求目標，但集合眾多同類型的個別低廉組件，將之整合為一，取代原來所需之巨大組件，如此既可滿足需求目標、又可降低資源耗費。

6. **原理名稱**：以羊易牛

直觀含義：

在容許的目標下，以較便宜的資源取代較貴的資源。

適用問題：

在發明過程中，面臨到的難題是：某些組件的複雜度、或成本極高，造成主體也必須變得更複雜、或成本也必須跟著提高，如此，將耗費極大的資源。

運用說明：

「以羊易牛」發明原理的基本精神在於，首先確認複雜度、或成本極高的組件。在需求目標得以容忍的範圍下，降低組件的複雜度、或成本。進而得以讓整體變得更簡單、或是以更低的資源達成整體需求目標。

7. **原理名稱**：堆金疊玉

直觀含義：

結合不同類型的東西以增強能力。

適用問題：

在發明過程中，面臨到的難題是：主體由單一類型的組件所構成，若要滿足多樣化、或是高品質的需求目標，則必須讓此單一組件具備滿足所有需求目標的能力。通常越是要能夠具備「一對多」的能力，其耗費的資源也就越龐大。

運用說明：

「堆金疊玉」發明原理的基本精神在於，首先將會耗費極大資源的需求目標分割成幾個子項目，再各自設計能夠滿足各對應子項目的不同類型組件。由於需求被簡化，所以各組件也不會耗用過多資源。最後，再將這些各個不同類型組件合而為一單個組件，如此既可滿足需求目標、又可降低資源耗損。

8. **原理名稱**：因地制宜

直觀含義：

根據區域特性，選擇適宜的對策方法。

適用問題：

在發明過程中，面臨到的難題是：構成主體的各個組件皆不相同，同時各組件的需求目標也不一致。若要以單一個設計來滿足各組件的不同需求目標，則會因為此設計需要同時涵蓋各組件需求，造成大量的資源耗費。

運用說明：

「因地制宜」發明原理的基本精神在於，首先根據主體的需求目標，切分出可被分離的個別需求目標。在不影響主體的需求目標下，僅以滿足此個別部分做為設計目標。而其他部分，也以滿足其對應需求做為設計目標。換言之，此設計原理將造成主體中的部分組件，在目標上並不一致，但是結合起來則仍可符合整體的需求目標。同時，由於滿足個別需求目標遠比必

須滿足所有需求目標容易，因此，又可大量減少資源浪費。

9. **原理名稱**：與世隔絕

直觀含義：

將不一樣的東西與其餘部分隔開。

適用問題：

在發明過程中，面臨到的難題是：主體中的某個組件明顯不同於其餘整體，而此特殊組件會對其餘部份造成影響，從而影響需求目標的達成。同時，由於此組件至關重要，對於整題而言無法移除。

運用說明：

「與世隔絕」發明原理的基本精神在於，首先將此特殊組件從整體中獨立出來。再來則根據此組件需求以及對整體的可能影響，尋求最適合的設計方式，解決其所可能造成之衝突。當此抽離組件被獨立設計完成後，再行將組件與主體其餘部分，根據需求目標重新整合設計之。

10. **原理名稱**：相反相成

直觀含義：

兩個完全相反的東西，實際上卻可以幫助對方。

適用問題：

在發明過程中面臨到的難題是：主體存在某些組件不可減化也無法移除，同時此組件所需滿足的需求目標，正好與其特質完全相反，若採用的設計欲直接滿足此需求目標，卻又極為耗費資源。

運用說明：

「相反相成」發明原理的基本精神在於，針對上述該組件，尋求與其特質相反的輔助體，將之加乘到該組件上。透過輔助體的相反特性幫助組件，如此即可降低資源的損耗，卻又可達成需求目標。

11. **原理名稱**：先出後入

直觀含義：

先移出再放回來。

適用問題：

在發明過程中面臨到的難題是：主體運作上，為了維持暫時無實際運作的組件，進而造成大量資源的耗費。

運用說明：

「先出後入」發明原理的基本精神在於，先找出這些暫時不需使用的組件，並將之停止運轉、甚或移出主體，直到再次需要這些組件時，再行重新啟動或移入，進而降低資源的使用。

12. **原理名稱**：移形換位

直觀含義：

換到另一個維度。

適用問題：

在發明過程中面臨到的難題是：主體中的某些組件，看似無法改變，但是卻又無法全然滿足新的需求目標。

運用說明：

「移形換位」發明原理的基本精神在於，針對看似無法改變的組件，換從另一個維度設計，使其不需改變主體太多，卻又可輕易滿足新的需求目標。

13. **原理名稱**：改頭換面

直觀含義：

與內涵無關，僅改變外在。

適用問題：

在發明過程中面臨到的難題是：不同主體的內部需求目標一樣，但是外部的介面組件卻因需求而有不同，若因此而必須修正各主體設計，則將造成不必要的浪費。

運用說明：

「改頭換面」發明原理的基本精神在於，盡可能切割主體的核心功能與外部介面組件關係。在不需修正主體下，僅設計外部介面組件，而滿足不同主體的需求目標。

14. **原理名稱**：化繁從簡

直觀含義：將複雜變為簡單。

適用問題：

在發明過程中面臨到的難題是：主體設計過於複雜，造成運作困難度增加或資源耗損。

運用說明：

「化繁從簡」發明原理的基本精神在於，先行確認過於複雜的組件，或是簡化其設計以降低運作的困難度，或是利用符合需求目標但成本遠為低廉的消耗性輔助體取代該組件，以降低資源耗損。或當外部組件簡化時，內部組件則可能變得更複雜。

15. **原理名稱**：改弦易轍

直觀含義：改變原來的方法。

適用問題：

在發明過程中面臨到的難題是：主體面對環境改變，不再能夠符合需求目標。

運用說明：

「改弦易轍」發明原理的基本精神在於，先行確認因變動所產生衝突而被影響的組件，改變這些組件的設計，進而使得主體在環境改變後仍能符合需求目標。

16. **原理名稱**：方枘圓鑿

直觀含義：

利用圓與方的明顯不同，幫助分辨方向或角度。

適用問題：

在發明過程中，面臨到的難題是：不同組件間的外觀、介面或操作過於一致或對稱，不利差異分辨，進而常常需要耗費更多資源分辨不同組件。

運用說明：

「方枘圓鑿」發明原理的基本精神在於，首先確認各組件的外觀、介面或操作特性，將原來一致性或對稱部分，轉化成不一致或不對稱。轉化後仍需滿足原組件需求目標，但卻可以輕易區分各組件關係。

17. **原理名稱**：兼容並蓄

直觀含義：

將各種不同的東西合而為一。

適用問題：

在發明過程中面臨到的難題是：需要多種不同的主體才能滿足需求目標。

運用說明：

「兼容並蓄」發明原理的基本精神在於，將上述這些不同主體設計成為組件，再行將這些組件構成單一主體，進而利用有限資源滿足需求目標。

18. **原理名稱**：一石二鳥

直觀含義：做一件可得兩個好處。

適用問題：

在發明過程中面臨到的難題是：主體設計必須滿足多項需求目標，若一一滿足之，則需要耗損大量資源。

運用說明：

「一石二鳥」發明原理的基本精神在於，確認會影響多項需求目標的關鍵組件，使其滿足兩種以上的需求；或是使此組件得以幫助其他組件，進而減少或降低主體的複雜度。

19. **原理名稱**：廢物利用

直觀含義：

將沒有用的產出物用來改善設計。

適用問題：

在發明過程中面臨到的難題是：主體運作過程中，會產生無用的東西。同時，為達成某些需求目標，主體卻會浪費更多資源。

運用說明：

「廢物利用」發明原理的基本精神在於，嘗試將產生的無用產出物，轉換幫助主體達成特定需求目標，進而減少資源浪費。

20. **原理名稱**：棄短就長

直觀含義：捨棄短處，選用長處。

適用問題：

在發明過程中面臨到的難題是：主體雖可滿足需求目標，但當需求目標提升時，主體就不再能夠滿足需求目標。

運用說明：

「棄短就長」發明原理的基本精神在於，將主體中可能因需求目標提升後，就可能造成影響的組件，改換成雖然更耗資源、但也更具彈性的組件，以便主體能夠滿足提升後的需求目標。

21. **原理名稱**：一以貫之

直觀含義：

在不同的環境或挑戰下，都可以只依靠一種方式或東西。

適用問題：

在發明過程中，面臨到的難題是：在不同的環境下，主體皆需要存在。但是，若分別針對不同的環境，就發展一個對應的主體，那麼將耗費極大的資源。

運用說明：

「一以貫之」發明原理的基本精神在於，根據主體的需求目標，找出可以作為跨越環境障礙的一個輔助體，然後再在此輔助體上發展主體。即使環境多有變化，依靠對應的輔助體，主體的需求目標也可不受影響。

22. **原理名稱**：以柔克剛

直觀含義：利用柔性設計取代鋼性設計。

適用問題：

在發明過程中面臨到的難題是：主體為滿足需求目標，必須將非常複雜的步驟與應用邏輯以實體方式設計出來，如此將耗費大量資源。

運用說明：

「以柔克剛」發明原理的基本精神在於，將複雜的應用邏輯之實體設計，改成利用軟體技術來設計，進而既可滿足需求目標，又可降低資源浪費。

23. **原理名稱**：防患未然

直觀含義：

事先預防，避免將來可能發生的困擾。

適用問題：

在發明過程中，面臨到的難題是：初期不見得會發生的問題，卻可能於過程中，或因運作、或因偶然、或因環境等因素造成嚴重衝突，使得主體無法正常因應、或為了因應而需耗費大量資源。

運用說明：

「防患未然」發明原理的基本精神在於，找出運作過程中，可能會因各種因素而產生問題的組件或行為，發展其對應的預防機制。或是能讓問題無法發生、或是即使問題發生也能自動修復等，從而得以維持整體的需求目標。

24. **原理名稱**：以拖待變

直觀含義：先不解決問題，直到事情發生再說。

適用問題：

在發明過程中面臨到的難題是：為了解決所有設計上可能出現的衝突，將會大量耗費資源。

運用說明：

「以拖待變」發明原理的基本精神在於，針對某些出現機率較低的衝突，根本不去事先預防。當衝突發生時，才犧牲部份資源解決衝突。

25. **原理名稱**：或多或少

直觀含義：

或是額外多犧牲一些，藉此換取更好的需求目標；或是比份內更少用一些，藉此降低不必要的損耗。

適用問題：

在發明過程中，面臨到的難題是：在標準的作法與使用資源規範下，為達成需求目標，可能頗為曠日廢時；或是為了達成特定的需求目標，在現有作法下，常需耗用大量資源。

運用說明：

「或多或少」發明原理的基本精神在於，首先找出造成曠日廢時或是耗用大量資源的關鍵組件或做法。此時，可從額外多犧牲一些資源的角度，找出改善目前的關鍵組件或作法，進而達到降低曠日廢時的情況發生；或是從略為降低需求目標品質的角度，改變現有組件或做法，進而降低大量資源的耗用。

26. **原理名稱**：順水推舟

直觀含義：

借助已有的力量來達成目標。

適用問題：

在發明過程中，面臨到的難題是：為滿足某些需求目標，需要花費額外步驟進而造成資源浪費。

運用說明：

「順水推舟」發明原理的基本精神在於，首先找出這些額外步驟的工作內容。同時分析在整體步驟中，是否存在某些已有的其他步驟，可以順便執行這些工作。這些順便執行的工作既可以達成對應之需求目標，同時也不應影響其原有工作。

27. **原理名稱**：週而復始

直觀含義：

固定週期下，自動反覆執行。

適用問題：

在發明過程中，面臨到的難題是：某些狀況的出現無法預期，由於無法事先或即時回應這些狀況，結果可能危害到主體需求目標。或是經常性的回應狀況，造成組件耗損，品質下降，進而產生資源耗費的問題。

運用說明：

「週而復始」發明原理的基本精神在於，找出無法預期的可能危害狀況，將其對應的回應行為改為週期性自動回應、或是改變已是週期性自動回應行為的頻率。自動回應行為的週期性頻率與狀況發生頻率有關，透過設計出滿足發生頻率的週期性自動回應，可以有效達成需求目標。

28. **原理名稱**：伺機而動

直觀含義：

等待事情發生後，再面對問題回應挑戰。

適用問題：

在發明過程中，面臨到的難題是：為滿足需求目標，主體中存在某些會反覆出現的行為，但其發生週期無法預估。若採用固定週期方式處理此類行

為，則可能耗費過多資源。

運用說明：

「伺機而動」發明原理的基本精神在於，將此類無法預估週期的運作行為，改由事件發生時才驅動執行。同時，根據事件類型，必須先設計好其回應行為。當事件發生時，必須能夠主動啟動對應之回應行為。

29. **原理名稱**：截彎取直

直觀含義：

讓原本彎彎曲曲的路徑變成更直、更好走。

適用問題：

在發明過程中，面臨到的難題是：現有作法存在步驟過於繁雜、瑣碎、或是緩慢等現象，進而造成需要花費更多資源來達成需求目標。

運用說明：

「截彎取直」發明原理的基本精神在於，首先確認各動作中，過於繁雜、瑣碎、或是運作緩慢的步驟。或是將這些步驟加以簡化、或是合併成為一個單一步驟、或是找出加速的做法，進而可以因此而降低資源的耗費，同時仍可達成需求目標。

30. **原理名稱**：繞道而行

直觀含義：

往前直走面臨阻礙時，換另一條路走。

適用問題：

在發明過程中，面臨到的難題是：雖然存在明確而直接的作法，但是此作法卻會耗用過多資源、甚至影響需求目標。

運用說明：

「繞道而行」發明原理的基本精神在於，調整原先運作步驟的先後順序、或加入某些額外的步驟，讓原先會產生的資源耗用、或需求目標難以達成的

問題，因為改變原步驟順序而消失。

31. **原理名稱**：移東就西

直觀含義：

若不能移船就岸，那就移岸就船。

適用問題：

在發明過程中面臨到的難題是：某些組件原來應該去配合主體的設計，但是這樣的做法卻會造成資源的損耗。

運用說明：

「移東就西」發明原理的基本精神在於，若是讓組件去配合主體會造成更大的資源損耗時，反之若可以改變主體的設計來遷就組件，同時又可降低資源損耗，那麼就不是設計組件以配合主體，而是設計主體以配合組件。

32. **原理名稱**：先斷後聞

直觀含義：

事先處理，以後再享受。

適用問題：

在發明過程中，面臨到的難題是：主體運作過程中，常常需要及時準備某些步驟所需用到的資源、或是在步驟之前必須先行執行某些運作，而上述這些行為卻因效率不佳或過度耗費資源而影響該步驟。同時，這些所需資源或運作卻又無法將之簡化或移除。

運用說明：

「先斷後聞」發明原理的基本精神在於，首先確認此步驟所需的資源或其前置運作，然後事先準備妥當這些資源或執行這些前置運作。等到執行至該步驟時，即可不需再等待而直接繼續執行。

33. **原理名稱**：釜底抽薪

直觀含義：

不從枝微末節下手，而是從根本上解決問題。

適用問題：

在發明過程中，面臨到的難題是：主體複雜度極高，極為耗費資源或是其效益不彰。若要改善各個組件以達成需求目標，卻又會更為耗費資源。

運用說明：

「釜底抽薪」發明原理的基本精神在於，不從改善各個組件著眼，而是從整體本身思考，找出高複雜度處、或資源耗費的原因，進而直接找出改善作法。

34. **原理名稱**：無中生有

直觀含義：將原本不存在的東西創造出來。

適用問題：

在發明過程中面臨到的難題是：主體中的某個組件極為耗費資源，甚至無法真實存在，但是若缺此組件，則主體又無法達成需求目標。

運用說明：

「無中生有」發明原理的基本精神在於，將此類組件分離出來後，以虛擬化的方式創造出來。而主體與此虛擬組件間，必需提供銜接介面。

35. **原理名稱**：見風轉舵

直觀含義：

利用已發生的情形，修正目前的做法。

適用問題：在發明過程中面臨到的難題是：主體的運作在面對不同問題時，其表現無法完全符合需求目標，甚至與目標需求越行越遠。

運用說明：

「見風轉舵」發明原理的基本精神在於，利用主體運作過程中所產生的相關資訊，配合需求目標調整主體的下一次運作。

36. **原理名稱**：扭轉乾坤

直觀含義：完全改變。

適用問題：

在發明過程中面臨到的難題是：主體的設計無論如何，均無法滿足需求目標。

運用說明：

「扭轉乾坤」發明原理的基本精神在於，當尋求各種方法皆無法滿足需求目標時，此時唯有跳脫現有的思考模式與框架，完全改變，重新發明所需的組件或主體。

3.2 TRIZ 比較

表 3-1 為數位發明原理與 TRIZ 40 條發明原理的對照表。左欄位是數位發明原理的名稱，右欄位則是對應到參考文獻[28]中 TRIZ 的發明原理編號。參考文獻[28]中，每一項發明原理底下都有相對應的解釋，而解釋的編號依序為 A, B, C, …。舉例來說，表 3-1 內「層次分明」所對照的發明原理為 P1-A、P1-B、P1-C 和 P15-B，意思是說，層次分明與參考文獻[28]中 TRIZ 的發明原理編號第 1 條"複製"的 A 解釋，將一個物件分割為幾個獨立的部份。和 B 解釋，讓分割一個物件變得很容易。和 C 解釋，增加分割或分裂的程度。發明原理編號第 15 條"動態"的 B 解釋，讓一個物件分割為好幾個部份，每個部份都與其它部分保有動態相關的性質。代表同樣的意義。

表 3-1 數位發明原理與 TRIZ 40 條發明原理對照表

數位發明原理	TRIZ 發明原理
照貓畫虎	ALL
層次分明	P1-A、P1-B、P1-C、P15-B
抓大放小	P31-A
執兩用中	P24-A、P30-B
化整為零	P5-A、P5-B、P33-A
以羊易牛	P14-B、P26-A、26-B、26-C、P27-A、P28-A、P29-A、P31-B、P37-A、P37-B、P39-A
堆金疊玉	P37-A、P37-B、P38-D、P39-B、P40-A

因地制宜	P3-B、P3-C
與世隔絕	P2-A
相反相成	P8-A、P8-B、P9-A、P9-B、P13-A
先出後入	P34-A、P34-B
移形換位	P17-A、P17-B、P17-C、P17-D、38-C
改頭換面	P32-A、P32-B
化繁從簡	P26-A
改弦易轍	P13-B、P13-C、P15-C、P28-C、P30-A、P35-A、P35-B、P35-C、P35-D
方枘圓鑿	P3-A、P4-A、P4-B
兼容並蓄	P6-A、P24-B、P28-D、P33-A
一石二鳥	P6-A
廢物利用	P22-A、P22-B、P25-B
棄短就長	P14-B、26-B、26-C、P28-A、P29-A、P31-B、P37-A、P37-B、P38-A、P38-B、P38-E、P39-A
一以貫之	P7-A、P7-B
以柔克剛	
防患未然	P11-A
以拖待變	P22-C
或多或少	P16-A、P22-C
順水推舟	P20-A、P22-A、P22-B、P25-A、P36-A
週而復始	P18-A、P18-B、P18-C、P19-A、P19-B、P20-A
伺機而動	P19-C
截彎取直	P20-B、P21-A
繞道而行	P14-A、P14-C
移東就西	P12-A
先斷後聞	P10-A、P10-B
釜底抽薪	P31-A
無中生有	
見風轉舵	P15-A、P23-A、P23-B、P28-B
扭轉乾坤	P18-D、P18-E

照貓畫虎比較特別，照貓畫虎可以對照整個 TRIZ 的發明原理，照貓畫虎的基本概念其實就是利用同樣的模式解決不同領域的問題。

P1-A、P1-B、P1-C 和 P15-B 都有透過將原本的主體分割來解決問題的意思。與層次分明為了避免牽一法動全身，而根據整體內部彼此關係密切的組

件，一塊塊分割出來，意思都是相同。所以層次分明可以對照 P1-A、P1-B、P1-C 和 P15-B。

P31-A 有去除造成困擾組件的意思。與抓大放小找出會產生衝突的組件，轉移、降低、甚至去除其對於整體的重要性，解決問題的方法都是相同。所以抓大放小可以對照 P31-A。

P24-A 和 P30-B 都有利用一個中介物質來解決或是隔離問題的意思。與執兩用中為了避免並存的衝突，利用加入另一新組件(輔助體)做為兩方的中介者，乃可降低合作兩方的耦合關係，意思都是相同。所以執兩用中可以對照 P24-A 和 P30-B。

P5-A、P5-B 和 P33-A 都有將物件結合起來，使用結合後一起運作的能力來改善問題的意思。與化整為零集合眾多同類型的個別低廉組件，將之整合為一，取代原來所需之巨大組件，解決問題的方法都是相同。所以化整為零可以對照 P5-A、P5-B 和 P33-A。

P14-B、P26-A、26-B、26-C、P27-A、P28-A、P29-A、P31-B、P37-A、P37-B 和 P39-A 都有使用某項物件取代原本的物件的意思。與以羊易牛在需求目標得以容忍的範圍下，降低組件的複雜度、或成本，都是更改了某項物件或目標。所以以羊易牛可以對照 P14-B、P26-A、26-B、26-C、P27-A、P28-A、P29-A、P31-B、P37-A、P37-B 和 P39-A。

P37-A、P37-B、P38-D、P39-B 和 P40-A 都有在原本的基礎上為了某項新的需求，而附加了可以解決這項需求的物件的意思。與堆金疊玉將各個不同類型組件合而為一單個組件，使用的方式都一樣。所以堆金疊玉可以對照 P37-A、P37-B、P38-D、P39-B 和 P40-A。

P3-B 和 P3-C 都有讓主體中每一個組件的功能都不一樣但都能符合該組建所需面對的需求的意思。與因地制宜滿足個別組件需求做為設計目標，在意義上相同。所以因地制宜可以對照 P3-B 和 P3-C。

P2-A 有將會造成困擾的組件抽離的意思。與與世隔絕將特殊組件從整體中獨立出來，運用方式都一樣。所以與世隔絕可以對照 P2-A。

P8-A、P8-B、P9-A、P9-B 和 P13-A 都有結合一個能夠解決問題的組件的意思。與相反相成尋求與其特質相反的輔助體，將之加乘到該組件上。透過輔助體的相反特性幫助組件，運用方式都一樣。所以相反相成可以對照 P8-A、P8-B、P9-A、P9-B 和 P13-A。

P34-A 和 P34-B 都有在運作行為中會轉換運作的模式的意思。與先出後入先找出主體內暫時不需使用的組件，並將之停止運轉、甚或移出主體，直到再次需要這些組件時，再行重新啟動或移入，都有轉變運作的模式，精神上是一樣。所以先出後入可以對照 P34-A 和 P34-B。

P17-A、P17-B、P17-C、P17-D 和 P38-C 都有從原本的基礎上增加另一個使用的維度的意思。與移形換位換從另一個維度設計，設計方式都是一樣。所以移形換位可以對照 P17-A、P17-B、P17-C、P17-D 和 P38-C。

P32-A 和 P32-B 都有改變外觀的意思。與改頭換面僅設計外部介面組件，而滿足不同主體的需求目標，作法都一樣。所以改頭換面可以對照 P32-A 和 P32-B。

P26-A 有使用簡單組件取代複雜組件的意思。與化繁從簡先行確認過於複雜的組件，或是簡化其設計以降低運作的困難度，或是利用符合需求目標但成本遠為低廉的消耗性輔助體取代該組件，意義都是相同。所以化繁從簡可以對照 P26-A。

P13-B、P13-C、P15-C、P28-C、P30-A、P35-A、P35-B、P35-C 和 P35-D 都有某些組件不符合原本的需求，進而改變以求達到目標的意思。與改弦易轍先行確認因變動所產生衝突而被影響的組件，改變這些組件的設計，進而使得主體在環境改變後仍能符合需求目標，意義都是相同。所以改弦易轍可以對照 P13-B、P13-C、P15-C、P28-C、P30-A、P35-A、P35-B、P35-C 和 P35-D。

P3-A、P4-A 和 P4-B 都有將原本規制與平衡的物件，改為不規制與不平衡的意思。與方枘圓鑿將原來一致性或對稱部分，轉化成不一致或不對稱，作法都一樣。所以方枘圓鑿可以對照 P3-A、P4-A 和 P4-B。

P6-A、P24-B、P28-D 和 P33-A 都有將原本不同主體的功能合在同一個主體

上的意思。與兼容並蓄將不同主體設計成為組件，再行將這些組件構成單一主體，作法都相同。所以兼容並蓄可以對照 P6-A、P24-B、P28-D 和 P33-A。

P6-A 有讓一個組件同時有多項功能的意思。與一石二鳥確認會影響多項需求目標的關鍵組件，使其滿足兩種以上的需求，運用方式相同。所以一石二鳥可以對照 P6-A。

P22-A、P22-B 和 P25-B 都有轉化原本無用或是有害的物件的意思。與廢物利用嘗試將產生的無用產出物，轉換幫助主體達成特定需求目標，運用方式相同。所以廢物利用可以對照 P22-A、P22-B 和 P25-B。

P14-B、26-B、26-C、P28-A、P29-A、P31-B、P37-A、P37-B、P38-A、P38-B、P38-E 和 P39-A 都有利用較好的組件取代原本無法符合需求的組件的意思。與棄短就長將主體中可能因需求目標提升後，就可能造成影響的組件，改換成雖然更耗資源、但也更具彈性的組件，使用方式相同。所以棄短就長可以對照 P14-B、26-B、26-C、P28-A、P29-A、P31-B、P37-A、P37-B、P38-A、P38-B、P38-E 和 P39-A。

P7-A 和 P7-B 都有讓一個主體配合另一個組件，以求滿足目標的意思。與一以貫之根據主體的需求目標，找出可以作為跨越環境障礙的一個輔助體，然後再在此輔助體上發展主體，在與組件互動配合的方式相同。所以一以貫之可以對照 P7-A 和 P7-B。

P11-A 有針對會發生的問題，準備應對手段的意思。與防患未然發展其對應的預防機制，運作的方式相同。所以防患未然可以對照 P11-A。

P22-C 有放任加大衝突，隨遇而安的意思。與以拖待變針對某些出現機率較低的衝突，根本不去事先預防。當衝突發生時，才犧牲部份資源解決衝突，意思相同。所以以拖待變可以對照 P22-C。

P16-A 和 P22-C 都有做更多或做更少來達成目標的意思。與或多或少額外多犧牲一些資源的角度，找出改善目前的關鍵組件或作法。或是從略為降低需求目標品質的角度，改變現有組件或做法。運作的方式相同。所以或多或少可以對照 P16-A 和 P22-C。

P20-A、P22-A、P22-B、P25-A 和 P36-A 都有在既定的運作行為下，執行某些運作的意思。與順水推舟同時分析在整體步驟中，是否存在某些已有的其他步驟，可以順便執行這些工作，運作的方式相同。所以順水推舟可以對照 P20-A、P22-A、P22-B、P25-A 和 P36-A。

P18-A、P18-B、P18-C、P19-A、P19-B 和 P20-A 都有反覆動作的意思。與週而復始透過設計出滿足發生頻率的週期性自動回應，運作的方式相同。所以週而復始可以對照 P18-A、P18-B、P18-C、P19-A、P19-B 和 P20-A。

P19-C 有在某一個時間點執行的意思。與伺機而動事件發生時才驅動執行，運作的方式相同。所以伺機而動可以對照 P19-C。

P20-B 和 P21-A 都有加速運作行為的意思。與截彎取直步驟加以簡化、或是合併成為一個單一步驟、或是找出加速的做法，運作的方式相同。所以截彎取直可以對照 P20-B 和 P21-A。

P14-A 和 P14-C 都有改變原本運作，繞過障礙的意思。與繞道而行調整原先運作步驟的先後順序、或加入某些額外的步驟，以求滿足需求，避免衝突的方式都一樣。所以繞道而行可以對照 P14-A 和 P14-C。

P12-A 有改變主體位置，以求滿足需求的意思。與移東就西設計主體以配合組件，從改變主體配合的角度來看都一樣。所以移東就西可以對照 P12-A。

P10-A 和 P10-B 都有為了避免物件拖延，而事先執行的意思。與先斷後聞事先準備妥當這些資源或執行這些前置運作。等到執行至該步驟時，即可不需再等待而直接繼續執行，運作模式都相同。所以先斷後聞可以對照 P10-A 和 P10-B。

P31-A 有將原本主體會產生問題的組件丟棄的意思。與釜底抽薪不從改善各個組件著眼，而是從整體本身思考，找出高複雜度處、或資源耗費的原因，進而直接找出改善作法，都是不耗資源解決表面問題，直接面對問題的核心。所以釜底抽薪可以對照 P31-A。

P15-A、P23-A、P23-B 和 P28-B 都有會根據輸入的不同而改變作法的意思。與見風轉舵利用主體運作過程中所產生的相關資訊，配合需求目標調整主

體的下一次運作，都是回饋機制的運用。所以見風轉舵可以對照 P15-A、P23-A、P23-B 和 P28-B。

P18-D 和 P18-E 都有改變原本基礎的作法，改採用原全不一樣的作法的意思。與扭轉乾坤跳脫現有的思考模式與框架，完全改變，重新發明所需的組件或主體，作法相同。所以扭轉乾坤可以對照 P18-D 和 P18-E。

其他尚有兩條數位發明原理，TRIZ 無法對照，分別是以柔克剛和無中生有。以柔克剛和無中生有都是因應數位技術發展的數位發明原理，強調的是利用虛擬化和軟體化的數位技術解決問題，Altshuller 所以歸納案例的時代，數位技術還未像現在如此蓬勃發展，所以無法在 TRIZ 的發明原理當中找對可以對照的概念。

而其中 P14-B、P26-B、P26-C、P28-A、P29-A、P31-B、P37-A、P37-B 和 P39-A 分別都可以對照以羊易牛與棄短就長，這是因為無法判定是否是為了節省成本為主要目標，而降低品質，還是為了提升品質為主要目標，而不在乎成本。所以上述這些發明原理就分別可以對照以羊易牛與棄短就長。

總結而言，數位發明原理完全可以涵蓋 TRIZ 的 40 條發明原理，使用者可以利用此三十六條原理來思考數位創新。下一章，我們將利用 RAID 的發展過程，來說明如何數位發明原理解決 RAID 的創新挑戰。

第四章 案例探討-RAID

第三章主要在探討在數位創新的發明原理。本章將以一個真實案例，RAID 發展的過程，來說明如何運用數位創新的發明原理。首先於 4.1 節中介紹 RAID 的基本概念。4.2 節則說明如何運用數位發明原理，發展出各種等級的 RAID。4.3 節將比較各種 RAID 的成本、容錯和效能。

4.1 RAID 介紹

RAID（Redundant Array of Inexpensive Disks，獨立磁碟冗餘陣列，台灣翻為"磁碟陣列"）是由美國加州大學柏克萊分校的 David A. Patterson、Garth A. Gibson 和 Randy Katz 在 1987 年共同發表的"A Case for Redundant Arrays of Inexpensive Disks (RAID)"論文中所提出的名詞[10][22][23][32]。

在論文發表的八十年代，第二儲存裝置(secondary storage)效能的發展速度與 CPU 和記憶體(memory)的發展速度相比，實在太慢，已成為電腦整體效能無法提升的主要原因之一 [20][21]。因此 RAID 的目的是用來取代單一、大容量、昂貴和效能高的硬碟(Single Large Expensive Disks, SLED)。RAID 的方法也讓第二儲存裝置的執行效能，能夠追上 CPU 與記憶體的發展速度。而 RAID 另一個比 SLED 更好的原因是，RAID 的容錯(fault-tolerant)能力。在這個議題上，一直是 RAID 發展以來討論的重點之一。而巢狀式 RAID(Nested RAID)是目前 RAID 議題中，討論的另一個重點。利用不同 RAID Level 的組合，來追求成本、容錯和效能之間平衡的解決方案。

4.2 發明原理應用

1. **問題描述**：

   在 80 年代中，第二儲存裝置效能的發展速度跟不上 CPU 與記憶體，已成為拖慢電腦整體效能的主要原因之一。同時硬碟成本昂貴、容量又低，也造成使用者頗大的負擔。

   問題分析：

   在電腦「主體」中，第二儲存裝置主要是由硬碟所組成。早期的硬碟「組

件」容量小、成本高，是頗為耗費「資源」的一種組件。

採用原理：

根據問題分析，使用「化整為零」數位發明原理。首先確認電腦「主體」中，使用 SLED 是造成硬碟「組件」設計複雜度和開發成本太高的原因。因此，集合多個容量小但低價位的硬碟，將之整合為一，取代原來所需之 SLED「組件」。從硬碟容量和硬碟價格的比例來看，硬碟容量小的硬碟比 SLED 低，但組合起來的容量卻比 SLED 大，而建置成本可比 SLED 還低。如此既可滿足儲存資料的「需求目標」、又可降低成本「資源」的耗費。而上述這樣的發明，是 RAID 的基本原型 MSID (Multiple Small Inexpensive Disks)。

數位發明案例：MSID Solution

2. **問題描述**：

MSID 若使用不同廠牌的硬碟，會因為各廠牌硬碟設計不同，造成集合在一起時，無法讀取到某些廠牌硬碟的資料。同時，因為成本或某些廠牌會停產，MSID 無法要求組成硬碟都是同一廠牌。

問題分析：

當組成 MSID 為「需求目標」時，會存在無法要求所有硬碟都是同一廠牌，但不是同一廠牌又會產生問題，這種魚與熊掌不可兼得、卻又必須並存的衝突。如果要求使用同一廠牌，當這個廠牌停產，會有 MSID 瓦解，彈性不足的問題。若是使用不同廠牌，又會有無法讀取到某些廠牌硬碟，運作失衡與不易維護的問題。進而最終無法達成使用 MSID 的「需求目標」。

採用原理：

根據問題分析，使用「執兩用中」數位發明原理。首先不管使用同一廠牌還是不同廠牌都會產生問題的原因是，由於組成 MSID 的硬碟是直接連結互動。此時，利用加入另一新組件做為兩方的中介者，進而產生 SCSI 這個「組件」。降低硬碟間的耦合關係，避免直接互動產生的問題，改透過

SCSI 這種間接互動的方式。

數位發明案例：SCSI

3. **問題描述**：

硬碟是用來儲存資料的重要裝置，因為資料很重要才會需要保存，所以如果因為硬碟損壞或是資料錯誤造成正確的資料遺失，將會造成極為嚴重的後果。MSID 沒有在硬碟損壞或是資料錯誤，將正確資料回復的機制，這存在著很大的風險。

問題分析：

當資料儲存於「主體」MSID 中，初期因為硬碟品質良好所以不會發生資料遺失的問題。但隨著時間的經過，因為硬碟品質有所下降的因素， 造成硬碟損壞或是資料錯誤，而若要回復正確資料，將耗費大量成本「資源」。

採用原理：

根據問題分析，使用「防患未然」數位發明原理。MSID 的運作過程中，會因硬碟品質下降的因素而產生資料遺失的問題，發展事先為正確的資料準備一份備份資料，也就是鏡射(mirror)的預防機制，進而產生 RAID Level 1 這個「主體」發明。舉例來說，當其中之一顆硬碟損壞或是資料錯誤造成正確的資料遺失時，還有正確的資料在另一顆硬碟當備份，可以讓正確的資料回復。所以當資料遺失時，還是得以維持使用正確的「需求目標」。

圖 4-1 為 RAID Level 1，又稱為”Mirrored set without parity”或”Mirroring”。圖中兩顆硬碟的顏色不一樣，但內容都一樣，代表儲存的檔案是同一份檔案，只是分別儲存於兩顆硬碟。利用鏡射(mirror)來達到容錯機制的磁碟陣列。組成 RAID Level 1 的硬碟都是互為鏡射。資料寫入是將同一份資料分別儲存進 RAID 中的各個硬碟，每份資料都具兩份。鏡射的因素，所以硬碟使用率是所有 RAID 中最低的。沒有 data striped，所以單人使用，並不會加快讀取速度。但在多人使用的系統中，因為每份資料都有備份，所以反而讀取的速度會變快。

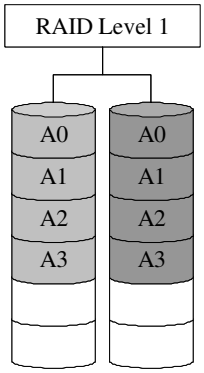

圖 4-1 RAID Level 1

數位發明案例：RAID Level 1

4. **問題描述**：

作業系統透過 DMAC(Direct Memory Access Controller, 直接記憶存取控制器)來管理第二儲存裝置。DMAC 原本是不支援 RAID。也不是所有使用者都需要使用 RAID，所以無法讓 DMAC 預設支援 RAID。而且讓 DMAC 負責 RAID 讀取和寫入資料與回復正確的資料，這些管理工作會造成 DMAC 的負擔，與增加 DMAC 設計上的難度。

問題分析：

若將 DMAC 視為「主體」，RAID 的管理工作視為「組件」。對於 DMAC 來說， RAID 的管理工作不屬於 DMAC 原本需要負責的工作，也不是所有使用者都需要 RAID 的功能，所以 RAID 的管理工作明顯不同於主體中其他的組件，讓 DMAC 預設支援 RAID 的管理工作，會造成 DMAC 設計的困難與整體效能的下降，從而影響節省成本「資源」與效能等需求目標的達成。同時，當使用者有建置 RAID 的需要時，RAID 的管理工作還是需要某個組件負責，對於 DMAC 而言還是無法移除。

採用原理：

根據問題分析，使用「與世隔絕」數位發明原理。首先將 RAID 的管理工作從 DMAC 中獨立出來。再來根據管理 RAID 的「需求目標」，發展出 RAID Controller 來負責 RAID 的管理工作。當 RAID 的管理工作被獨立設

計由 RAID Controller 負責後，DMAC 要根據支援與 RAID Controller 互動這個需求目標重新設計。這樣 DMAC 面對 RAID 還是與面對一般第二儲存裝置一樣，繁雜的管理工作就交由 RAID Controller 負責，對於 DMAC 來說，設計複雜度與成本都不會大幅提升，但還是有辦法支援 RAID 的管理工作。

數位發明案例：RAID Controller

5. **問題描述**：

RAID Level 1 利用鏡射實作容錯機制，讓硬碟損壞或是資料錯誤造成正確的資料遺失時，可以回復正確的資料。但也因為鏡射的原因，每一份資料所需的硬碟空間都是沒有使用鏡射機制的兩倍，造成硬碟使用率太低只有50%的問題。

問題分析：

若將 RAID Level 1 視為「主體」，使用鏡射實作容錯機制可滿足當硬碟損壞時，回覆正確資料的需求目標，但硬碟使用率太低是一大問題，當「需求目標」提升為需有容錯機制，硬碟使用率又能提高時，RAID Level 1 就不再能滿足「需求目標」。

採用原理：

根據問題分析，使用「棄短就長」數位發明原理。「主體」發明 RAID Level 1 中，因「需求目標」提升為有容錯機制，硬碟使用率又能提高後，原本由鏡射實作的容錯機制，會是影響硬碟使用率無法提升的組件。改用漢明碼實作容錯機制的組件，雖然漢明碼設計的難度比鏡射高，但是可以滿足有容錯機制，硬碟使用率提高的「需求目標」。RAID Level 2 就是漢明碼實作容錯機制的「主體」發明。

圖 4-2 為 RAID Level 2，又稱為"Hamming code parity"。圖中硬碟的顏色不一樣，代表檔案不一樣，硬碟內容的下標顯示 ECC，代表這個內容是漢明碼錯誤檢查碼。採用漢明碼(Hamming code)來實作容錯機制的磁碟陣列。將資料以漢明碼的方式編碼後，等量分割儲存進 RAID。存入的資料因加

入漢明碼，所以會比原始資料大。使用漢明碼的方式回復資料，會比使用同位元檢查碼的方式回復資料，在硬體設計上較為容易。實作 RAID Level 2 的硬碟數取決於硬碟磁區的位元數。沒有 RAID Level 2 的商業應用，RAID Level 2 只存在於理論和實驗室，因為實作 RAID Level 2 所需的硬碟成本太高。與 RAID Level 1 做比較，增加了硬碟使用率。

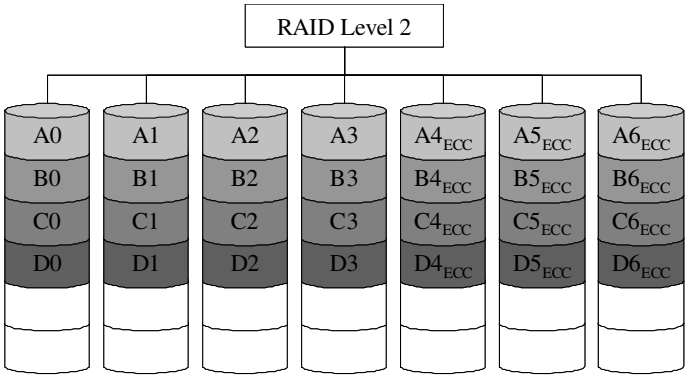

圖 4-2 RAID Level 2

數位發明案例：RAID Level 2

6. **問題描述**：

RAID Level 2 為了改善 RAID Level 1 硬碟使用率的問題，而採用漢明碼來實作容錯機制。但硬碟使用率與 SLED 還是不夠好。舉例來說，使用漢明碼實作容錯機制的 RAID Level 2，如果要使用 4 顆資料硬碟，需準備另外 3 顆硬碟來存錯誤檢查碼，就是 7 顆硬碟只能用到 4 顆硬碟。如果每顆硬碟是 100G 的容量，700G 的容量，使用者只能用到 400G。當然這個比例會依照硬碟數不同和設計不同而有不同數據產生，但是建置的成本高是可以預期的結果。

問題分析：

若將 RAID Level 2 視為「主體」，RAID Level 2 的容錯機制視為組件，以漢明碼實作的容錯機制會有實作成本太高的問題，造成 RAID Level 2 的成本也必須跟著提高。如此，將耗費極大的成本「資源」。

採用原理：

根據問題分析，使用「以羊易牛」數位發明原理。首先漢明碼實作的容錯機制是造成設成本提升的「組件」。在建置容錯機制與提高硬碟使用率的「需求目標」之下，使用低成本的同位元錯誤檢查碼來實作容錯機制，進而產生 RAID Level 3 這個「主體」發明。同位元錯誤檢查碼實作的容錯機制，不管組成 RAID 的硬碟有多少顆，儲存同位元錯誤檢查碼的硬碟一定只有一顆，這樣就可以以更低的成本「資源」達到建置容錯機制與硬碟使用率提高的「需求目標」。

圖 4-3 為 RAID Level 3，又稱為"Striped set with dedicated parity"或"bit interleaved parity"或"byte level parity"。圖中硬碟的顏色不一樣，代表檔案不一樣，硬碟內容的下標顯示 P，代表這個內容是同位元錯誤檢查碼。使用同位元檢查碼來實作容錯機制的磁碟陣列。將資料編碼，用位元組(byte)的方式等量分割儲存進 RAID，而錯誤檢查碼單獨存入一顆專門儲存錯誤檢查碼的硬碟。與 RAID Level 2 做比較，降低實作容錯機制的難度和增加了硬碟使用率。

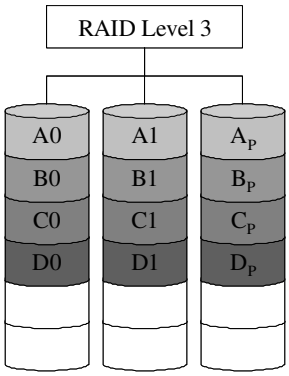

圖 4-3 RAID Level 3

數位發明案例：RAID Level 3

7. **問題描述**：

RAID Level 3 是以位元組(bytes)的方式將資料儲存進 RAID，所以每次讀取一組位元組的資料，就要到專門儲存錯誤檢查碼的硬碟，找出錯誤檢查

碼，做一次錯誤檢查。當資料越大，就執行越多次，影響讀取速度，進而造成浪費更多時間來達成資料快速讀取的目標。

問題分析：

RAID Level 3 這個「主體」的「行為」中，每次讀取一組位元組的資料，就要到專門儲存錯誤檢查碼的硬碟，找出錯誤檢查碼，做一次錯誤檢查的「步驟」，而這個「步驟」過於瑣碎是造成讀取速度緩慢的主因，進而造成浪費更多時間「資源」來達成資料快速讀取這個「需求目標」。

採用原理：

根據問題分析，使用「截彎取直」數位發明原理。將位元組(bytes)合併為位元組區塊，這個位元組區塊稱為磁區(block)，使用磁區的方式將資料儲存進 RAID，進而產生 RAID Level 4 這個「主體」發明。舉例來說，假設每個磁區的大小為 100 個位元組，要讀取一個 100 個位元組大小的資料，RAID Level 3 要讀取專門儲存錯誤檢查碼的硬碟 100 次，RAID Level 4 只要讀取 1 次，便可以減少時間的耗費。因此 RAID Level 4 這個「主體」，因為改用磁區後減少到專門儲存錯誤檢查碼的硬碟找出錯誤檢查碼，這個「步驟」的次數，加快資料讀取的速度，減少時間「資源」的耗費，但還是可以達成資料快速讀取這個「需求目標」。

圖 4-4 為 RAID Level 4，又稱為”Block level parity”。圖中硬碟的顏色不一樣，代表檔案不一樣，硬碟內容的下標顯示 PB，代表這個內容是同位元錯誤檢查碼，並以磁區的方式儲存。與 RAID Level 3 相同，也是將同位元檢查碼存入一顆專門儲存錯誤檢查碼的硬碟，但 RAID Level 4 的資料編碼後，是以磁區(block)的方式將資料等量分割儲存進 RAID。RAID Level 4 以磁區的方式分割資料，是期望在讀寫資料的效能上比 RAID Level 3 好。與 RAID Level 3 做比較，RAID Level 4 在讀取和寫入隨機資料的速度上比較好。

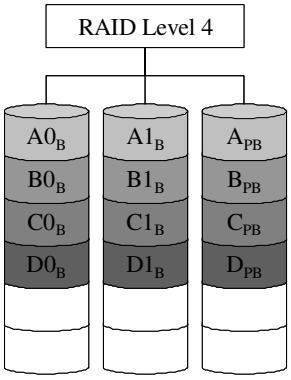

圖 4-4 RAID Level 4

數位發明案例： RAID Level 4

8. **問題描述**：

RAID Level 4 的儲存錯誤檢查碼的硬碟，因為每次做資料讀取和寫入都必須要參與運作，造成儲存錯誤檢查碼硬碟的硬碟損壞率比儲存資料的硬碟高。

問題分析：

若將 RAID Level 4 視為「主體」，儲存錯誤檢查碼的硬碟視為「組件」。因為儲存錯誤檢查碼的硬碟，每次做資料讀取和寫入都必須要參與運作，造成硬碟損壞率較高，進而耗費成本「資源」。若要改善儲存錯誤檢查碼硬碟的品質，以達成降低硬碟損壞率的「需求目標」，卻又會更為耗費成本「資源」，且不一定又改善的效果。

採用原理：

根據問題分析，使用「釜底抽薪」數位發明原理。不從改善儲存錯誤檢查碼硬碟品質著眼，而是從整體 RAID Level 4 本身思考。使用單一的硬碟來儲存錯誤檢查碼，造成這顆硬碟運作頻繁，是造成損壞率很高的原因。將錯誤檢查碼平均分散到組成 RAID 的硬碟，也平均分散硬碟損壞的風險，這樣就不會有單獨一顆硬碟特別容易損壞的問題。而 RAID Level 5 就是利用這個概念的「主體」發明。

圖 4-5 為 RAID Level 5，又稱為”Striped set with distributed parity”

或”interleave parity”。圖中硬碟的顏色不一樣，代表檔案不一樣，硬碟內容的下標顯示 PB，代表這個內容是同位元錯誤檢查碼，並以磁區的方式儲存。資料分割儲存的方式與 RAID Level 4 相同，但 RAID Level 5 不使用獨立的硬碟儲存同位元檢查碼，而是將同位元檢查碼分配儲存進組成 RAID 的各硬碟中。與 RAID Level 3 和 RAID Level 4 做比較，RAID Level 5 讀取和寫入隨機資料的速度上比較好，而讀寫連續資料的效能上比 RAID 3 不相上下。RAID 5 是公認成本、容錯和效能獲得平衡的解決方案。

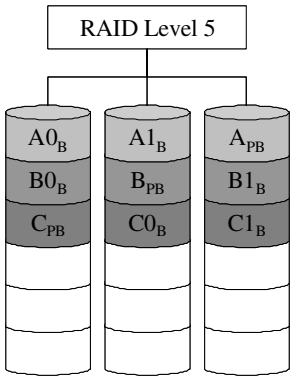

圖 4-5 RAID Level 5

數位發明案例：RAID Level 5

9. **問題描述**：

RAID Level 5 只允許同時損壞一顆硬碟，如果同時損壞兩顆硬碟，將會造成 RAID 瓦解和資料遺失的風險。

問題分析：

若將 RAID Level 5 視為「主體」，RAID Level 5 可滿足同時損壞一顆硬碟，而不使得 RAID 瓦解的「需求目標」，但當「需求目標」提升為同時損壞兩顆硬碟，而不使得 RAID 瓦解時，RAID Level 5 這個「主體」就不再能夠滿足新的「需求目標」。

採用原理：

根據問題分析，使用「棄短就長」數位發明原理。「主體」發明 RAID Level 5 中，因「需求目標」提升為同時損壞兩顆硬碟後，原本由同位元錯誤檢

查碼實作的容錯機制，會是影響同時損壞兩顆硬碟的組件。改用奇偶同位元錯誤檢查碼實作容錯機制的組件，雖然奇偶同位元錯誤檢查碼需更耗硬碟使用「資源」，但是可以同時損壞兩顆硬碟的「需求目標」。RAID Level 6 就是奇偶同位元錯誤檢查碼實作容錯機制的「主體」發明。

圖 4-6 為 RAID Level 6，又稱為"Striped set with dual distributed parity"。圖中硬碟的顏色不一樣，代表檔案不一樣，硬碟內容的下標顯示 PB 和 QB，代表這個內容是奇偶同位元錯誤檢查碼，並以磁區的方式儲存。資料分割和儲存同位元的方式與 RAID Level 5 相同，不同的是，資料檢驗的方式使用奇偶同位元檢查碼，讓容錯能力大幅增加，可以同時允許兩顆硬碟損壞。與 RAID Level 5 做比較，增加了容錯的能力。

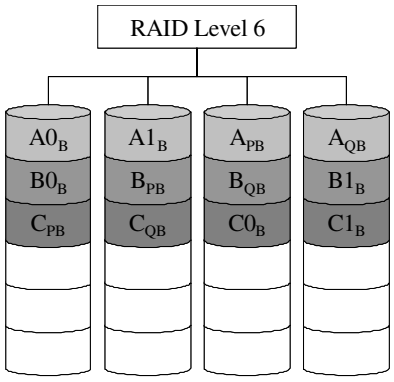

圖 4-6 RAID Level 6

數位發明案例：RAID Level 6

10. **問題描述**：

RAID 一路的發展都是在追求成本、容錯和效能之間平衡的解決方案。但是發展到 RAID Level 6，雖然容錯的能力越來越好，但效能卻是降低。

問題分析：

在 RAID Levels 標準需有容錯機制的作法與使用有限成本「資源」規範下，為達成效能提升的「需求目標」，可能頗為曠日廢時。

採用原理：

根據問題分析，使用「或多或少」數位發明原理。首先容錯機制是達成效能提升的「需求目標」曠日廢時的關鍵「組件」。從降低回復正確資料的功能以求提升效能，這個「需求目標」的角度，放棄容錯機制，產生 RAID Level 0 這個「主體」發明，進而因為不需要再進行錯誤檢查，使得效能提升，也降低大量計算「資源」的耗用。

圖 4-7 為 RAID Level 0，又稱為"Striped set without parity"或"Striped"。 圖中兩顆硬碟的顏色都是一樣，但內容不一樣，代表儲存的檔案是同一份檔案，只是分割儲存於兩顆硬碟。一個沒有容錯機制的磁碟陣列。資料會以等量分割(data striped)的方式儲存進 RAID。沒有容錯機制，所以任何一顆硬碟的損壞，將會造成所有資料的遺失。但也因沒有容錯機制，所以硬碟的使用率是所有 RAID 中最高的，基本上就是 100%。資料的讀取與寫入是所有 RAID 中速度最快的。在最初"A Case for Redundant Arrays of Inexpensive Disks (RAID)"這篇論文中是沒有 RAID Level 0，在查到的文獻中，最早出現這個名詞是在 1990 年同一批作者所發表的論文"An evaluation of redundant arrays of disks using an Amdahl 5890 "。雖然 RAID Level 0 沒有容錯機制，在嚴謹定義上，不能被稱為一個 RAID Level，但 RAID Level 0 常被用來與其他 RAID Level 組合與比較，就是 Nested RAID 的方式，所以為了稱呼上的方便，RAID Level 0 還是被稱為一個 RAID Level。

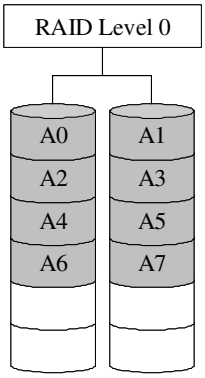

圖 4-7 RAID Level 0

數位發明案例：RAID Level 0

11. **問題描述**：

或許對於追求效能且不在乎資料遺失的使用者來說，RAID Level 0 是最佳的解決方案。但是對於大部份的使用者來說，還是不能沒有容錯機制。成本、容錯和效能都是選擇 RAID levels 的考量之一

問題分析：

若將 RAID Levels 視為主體，若要滿足成本、容錯和效能等多樣化的「需求目標」，則耗費的成本「資源」也就越龐大。

採用原理：

根據問題分析，使用「堆金疊玉」數位發明原理。首先要成本、容錯和效能兼具的「需求目標」將會耗費極大「資源」，將這些「需求目標」分割為成本、容錯或效能等子項目。再各自設計能夠滿足成本、容錯或效能，這些子項目的「組件」RAID Level。舉例來說，最能滿足效能與成本的 RAID Level 是 RAID Level 0。最能滿足容錯的 RAID Level 是 RAID Level 6。最後，將這些 RAID Level 合而為單個 RAID Level，如此可以滿足成本、容錯或效能兼具的需求目標，又可降低使用「資源」耗損。

下列的 RAID Levels 都屬於 Nested RAID，所以圖示的顏色與內容都是與組成的 RAID Levels 為基礎，代表的意義也是一樣。

圖 4-8 為 RAID Level 0+1，又稱為"RAID Level 01"，但容易與"RAID Level 1"產生渾淆。這類型的 RAID 組成的方式是，上層是 RAID Level 1，底層是 RAID Level 0。資料寫入先做鏡射，再分割儲存進 RAID。與 RAID Level 5 做比較，在效能和容錯的能力上都比較好，但成本 RAID Level 0+1 比較高。

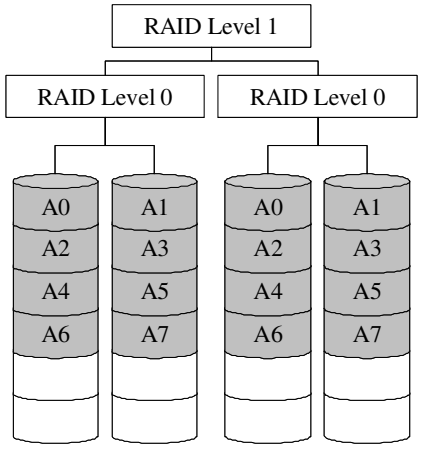

圖 4-8 RAID Level 0+1

圖 4-9 為 RAID Level 10，又稱"RAID Level 1+0"。這類型的 RAID 組成的方式是，上層是 RAID Level 0，底層是 RAID Level 1。資料寫入先做分割，再鏡射儲存進 RAID。容錯的能力比 RAID Level 0+1 好。

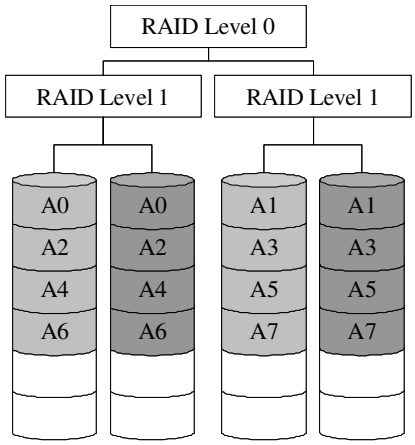

圖 4-9 RAID Level 10

圖 4-10 為 RAID Level 50，這類型的 RAID 組成的方式是，上層是 RAID Level 0，底層是 RAID Level 5。資料寫入先做鏡射，再以 RAID Level 5 的方式儲存進 RAID。與 RAID Level 5 做比較，增加了容錯與資料讀取和寫入的速度。

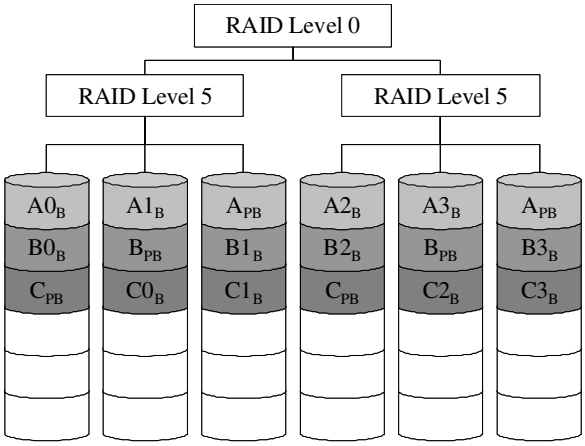

圖 4-10 RAID Level 50

圖 4-11 為 RAID Level 60，這類型的 RAID 組成的方式是，上層是 RAID Level 0，底層是 RAID Level 6。資料寫入先做鏡射，再以 RAID Level 6 的方式儲存進 RAID。與 RAID Level 6 做比較，增加了容錯與資料讀取和寫入的速度。可以同時損壞 4 顆硬碟，還保持運作的能力是本文所有介紹的 RAID Levels 中最好的。

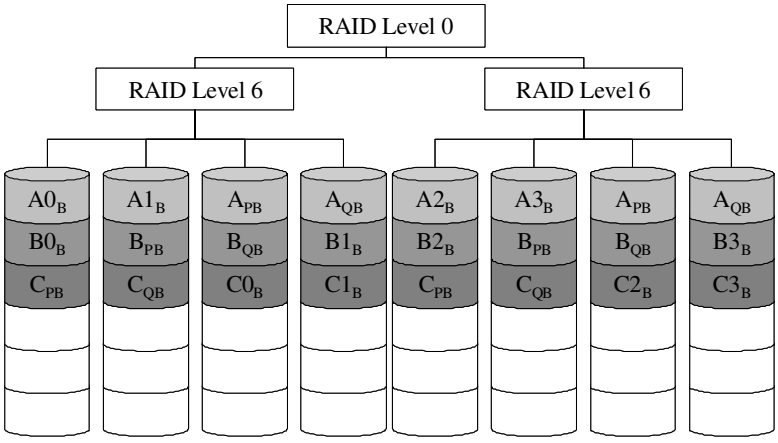

圖 4-11 RAID Level 60

數位發明案例：Nested RAID

12. **問題描述**：

要建置 Hardware RAID 需要購買昂貴的 RAID Controller，會造成使用者成本上的負擔。

問題分析：

若將電腦視為「主體」，第二儲存裝置視為「組件」，為了滿足在第二儲存裝置建置 RAID 的「需求目標」，需要購買昂貴的 RAID Controller 來建構與管理 RAID。建構與管理 RAID 的「步驟」與「應用邏輯」非常複雜，所以 RAID Controller 以實體方式設計出來，如此將耗費大量「成本」資源。

採用原理：

根據問題分析，使用「以柔克剛」數位發明原理。將 RAID Controller 複雜的應用邏輯之實體設計，改成利用軟體技術在虛擬概念上實作 RAID，進而產生軟體 RAID 這個「主體」發明。使用者在概念上還是滿足使用 RAID 的「需求目標」，但又可降低成本「資源」浪費。

Software RAID，由 CPU 負責 RAID 的管理工作。成本上來看，由軟體實作 RAID 會比硬體實作 RAID 便宜。開發上也會比較容易，因為少了去顧慮硬體品質的問題，複雜度會降低不少。總結來說，Software RAID 的好處是不必添購昂貴的 RAID Controller 就可以使用 RAID 所帶來的好處。但缺點是在運作的效能上無法與 Hardware RAID 相比。但如果使用者不在乎運作的效能，只希望能有 RAID 的容錯機制和低的 RAID 建置成本，Software RAID 會很符合使用者的需求。

數位發明案例：Software RAID

13. **問題描述**：

建置 RAID 的硬碟，為了確保運作效能的品質，通常都會選用同一廠牌、同一型號與同一出廠時間的硬碟。但經常會有同時損壞的顆數，超過 RAID Level 所允許範圍，產生 RAID 瓦解的問題。

問題分析：

建置 RAID 的硬碟「組件」，因為品質過於一致，造成損壞的時間也很一致，超過 RAID Level 所允許的顆數，而有 RAID 瓦解的問題，進而需要耗費更多「資源」，來確保硬碟的品質。

採用原理：

根據問題分析，使用「方枘圓鑿」數位發明原理。首先確認硬碟「組件」的品質，將原本同一廠牌、同一型號與同一出廠時間的硬碟，轉化為不同廠牌、不同型號或不同出廠時間的硬碟。轉化後還是滿足原本擔任組成 RAID 硬碟的需求目標，但卻會因為品質不一，而使得損壞時間也不一，不會有組成 RAID 的硬碟同時損壞的顆數，超過 RAID Level 所允許範圍，產生 RAID 瓦解的問題。

RAID 的容錯機制依設計的不同，允許同時間損壞的硬碟數量也不同。RAID Level 0 不允許硬碟損外是特例。RAID Level 1~5 都只允許同時間損壞一顆硬碟，RAID Level 6 允許同時間損壞兩顆硬碟。而以 RAID Level 0 為輔助的 Nested RAID，雖然總體允許損壞的硬碟數較多，但每個子陣列所允許損壞硬碟數還是與原來的 RAID Level 一樣。

數位發明案例：組成 RAID 的硬碟，要選擇品質不一制的硬碟

在上述 RAID 案例中，13 個數位發明案例共使用到 12 個數位發明原理。其中棄短就長在兩個案例中被使用。環顧整個 RAID 發展的過程，所面臨到會造成矛盾的問題，都可以使用數位發明原理來解決。所以數位發明原理的確可以幫助使用者於數位創新上。

4.3 RAID 比較

表 4-1、表 4-2 和表 4-3 分別針對各式 RAID 的成本、容錯與效能做比較。表 4-1 為 RAID Levels 成本比較表，內容與建置成本有關，數字越大成本越差。表 4-2 為 RAID Levels 容錯比較表，內容與容錯能力有關，數字越大越好。表 4-3 為 RAID Levels 效能比較表，內容與效能有關，數字越大效能越好。

RAID Level 1 是最早出現的 RAID 產品，有容錯機制，比 SLED 還好。但是 RAID Level 1 因為使用鏡射實作容錯機制，造成硬碟使用率太低。而 RAID Level 2 則使用漢明碼實作容錯機制，改善 RAID Level 1 硬碟使用率太低的問題。但是 RAID Level 2 使用漢明碼實作容錯機制，會造成建置成本太高。RAID Level 3 則使用同位元錯誤檢查碼實作容錯機制，改善 RAID Level 2 建置

成本太高的問題。但是 RAID Level 3 的效能不好，因為資料是以位元組分割儲存至 RAID 硬碟。RAID Level 4 則使用磁區分割，改善 RAID Level 3 效能不好的問題。但是 RAID Level 4 進展到 RAID Level 5，是因為 RAID Level 4 使用專門的硬碟儲存錯誤檢查碼，會造成這顆硬碟損壞率特別高。RAID Level 5 則分散儲存錯誤檢查碼，改善 RAID Level 4 中專門儲存錯誤檢查碼的硬碟損壞率特別高的問題，這是比較表中無法看到的問題。但從表 4.3 效能比較表來看，RAID Level 5 也因為分散儲存錯誤檢查碼，所以在實際應用上，效能會比 RAID Level 4 好。RAID Level 5 的容錯機制，只允許損壞一顆硬碟，而 RAID Level 6 使用奇偶同位元錯誤檢查碼實作容錯機制允許損壞兩顆硬碟，改善 RAID Level 5 只允許損壞一顆硬碟的問題。RAID Level 6 雖然容錯能力越來越好，但是成本與效能卻往下降，RAID Level 0 則放棄容錯機制，改善 RAID Level 6 成本與效能下降的問題。但沒有容錯機制，由於 RAID Level 0 能有效增進效能，所以 RAID 0+1, 10, 50, 與 60，分別利用不同 RAID Level 與 RAID Level 0 組合，進而滿足 trade-off 的需求目標。

表 4-1 RAID Levels 成本比較表

RAID Level	需要幾顆硬碟	硬碟使用率	建構 RAID 的成本
0	≧2	100%	2
1	≧2	50%	4
2	依設計而不同	~ 70-80%	10
3	≧3	(N-1)/N	4
4	≧3	(N-1)/N	4
5	≧3	(N-1)/N	4
6	≧4	(N-2)/N	6
0+1	≧4+2N	50%	6
10	≧4+2N	50%	6
50	≧6+2N	(N-1)/N	8
60	≧8+2N	(N-2)/N	10

表 4-2 RAID Levels 容錯比較表

RAID Level	最多容許損壞幾顆硬碟數量	將資料回復正確的能力	在硬碟損壞下，繼續工作的能力	將資料回復正確的速度
0	不允許	沒有	沒有	沒有
1	1	8	7	8
2	1	4	6	6

3	1	6	6	2
4	1	6	6	2
5	1	6	4	2
6	2	9	4	3
0+1	每個子陣列 1 顆	8	8	7
10	每個子陣列 1 顆	9	9	8
50	每個子陣列 1 顆	7	6	4
60	每個子陣列 2 顆	10	5	6

表 4-3 RAID Levels 效能比較表

RAID Level	讀取隨機資料的速度	寫入隨機資料的速度	讀取連續資料的速度	寫入連續資料的速度
0	8	8	9	8
1	6	6	4	6
2	4	2	8	5
3	6	2	8	5
4	8	3	6	4
5	9	4	7	5
6	9	2	7	4
0+1	9	7	9	7
10	9	7	9	7
50	9	6	8	6
60	9	5	8	5

資料來源[1][14][30]

第五章 結論與未來研究方向

5.1 結論

基於 TRIZ 與數位技術特性，本論文提出一個數位創新的方法論。在此方法中，數位發明共由三十六條原理所構成。而這些數位發明原理，也可以完全涵蓋 TRIZ 的 40 發明原理。而同時，另有六條發明原理是 TRIZ 所無法支援，而原因則歸咎於數位技術的出現。為了驗證所提出的數位發明原理確實有助於數位創新，我們尚利用 RAID 發展過程中，當 RAID 發明遇到瓶頸時，說明如何使用數位發明原理來解決。

5.2 未來研究方向

(1) 建立數位發明案例庫。

依照 Altshuller 團隊的作法，收集目前世界上的產品或概念。歸納與整理這些產品運用到那一條數位發明原理。也藉此再次修正數位發明原理，以符合真實世界的現況。

(2) 結合數位發明案例庫與專利。

數位發明案例庫與世界知識產權組織(World Intellectual Property Organization, WIPO)做結合，讓使用者在解決問題的過程中得知目前有哪些專利與要解決的問題有相關，藉此避免或是使用某項專利。

參考文獻

1. Adaptec, "Which RAID level is right for me?," *Adaptec*, http://www.adaptec.com/NR/rdonlyres/874D145E-F64F-4804-9E27-037BC5A9DCE0/0/3994_RAID_WhichOne_v112.pdf
2. Altshuller, G., *The innovation algorithm: TRIZ, systematic innovation and technical creativity*, *Technical Innovation Centerr , Inc.*, 1999
3. Altshuller, G., Altov , H. and Shulyak, L., *And suddenly the inventor appeared: TRIZ, the theory of inventive problem solving*, *Technical Innovation Centerr , Inc.*, 1996
4. Altshuller, G., Shulyak, L., Rodman, S. and Shulyak, L., *40 principles: TRIZ keys to innovation*, *Technical Innovation Centerr , Inc.*, 1997
5. Beardon, C. and Beardon, C., *Digital creativity: a reader*, *Taylor and Francis*, 2002
6. Brown, A. and Patterson, D.A., "Towards availability benchmarks: A case study of software RAID systems," *Proc 2000 USENIX Annual Technical Conf.*, 2000, pp. 263-276
7. BuffaloState,"Why study creativity?," *International Center for Studies in Creativity*, http://www.buffalostate.edu/creativity/whycreativity.xml
8. Chang, H. T. and Chen, J. L., "Eco-Innovative Examples for 40 TRIZ Inventive principles," *The TRIZ Journal*, 2003
9. Chai, K. H., Zhang, J. and Tan K. C., "A TRIZ-Based Method for New Service Design," *Journal of Service Research*, Vol. 8, no. 1, 2005, pp. 48-66
10. Chen, P. M., Gibson, G. A., Katz, R. H. and Patterson, D. A., "An evaluation of redundant arrays of disks using an Amdahl 5890," *Proceedings of the 1990 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, 1990, pp.74-85
11. Dourson, S., "The 40 inventive principles applied to finance," *The TRIZ journal*, 2004
12. Hipple, J., "40 inventive principles with examples for chemical engineering,"

The TRIZ journal, 2005

13. Kelly, O., *Digital Creativity*, *Calouste Gulbenkian Foundation*, 1996
14. Kozierok, C. M., "Summary comparison of RAID levels," *The PC Guide*, http://www.pcguide.com/ref/hdd/perf/raid/levels/comp-c.html
15. Kressel, H. and Lento, T. V., *Competing for the Future: How Digital Innovations are Changing the World*, *Cambridge University Press*, 2007
16. Mann, D., "An introduction to TRIZ: the theory of inventive problem solving," *Creativity and Innovation Management*, Vol. 10,Issue 2, 2003, pp. 123-125
17. Marsh, D. G., Waters, F. H. and Marsh, T. D., "40 inventive principles with applications in education," *The TRIZ journal*, 2005
18. Mishra, U., "The revised 40 principles for software inventions," *trizsite journal*, 2006
19. Mishra, U., *TRIZ principles for information technology*, *Technical Innovation Center Inc.*, 2007
20. Moore, G. E., "Progress in digital integrated electronics," *Proc IEEE Digital Integrated Electronics Device Meeting*, 1975, pp. 11
21. Myers, G. J., Yu, A. Y. C. and House, D. L., "Microprocessor technology trend," *Proc IEEE*, Vol. 74, no. 12, 1986, pp. 1605-1622
22. Patterson D. A., Gibson, G. and Katz, R. H., "A case for redundant arrays of inexpensive disks (RAID)," *Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, 1988, pp.109-116
23. Patterson, D. A., Chen, P. M., Gibson, G. and Katz, R. H., "Introduction to redundant arrays of inexpensive disks (RAID)," *COMPCON Spring '89. Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage*, 1989, pp.112-117
24. Poppe, G. and Gras, B., "Triz in the process industry," *The TRIZ journal*, 2002
25. Retseptor, G., "40 inventive principles in quality management," *The TRIZ journal*, 2003
26. Salamatov, Y., *TRIZ: the right solution at the right time: a guide to innovative*

problem solving, *Insytec*, 1999

27. Savransky, S. D., *Engineering of creativity:introduction to TRIZ methodology of inventive problem solving*, *CRC Press*, 2000.
28. Tate, K. and Domb, E., "40 Inventive Principles with Examples", *The TRIZ Journal*, 1997
29. Terninko, J., Zusman, A. and Zlotin, B., *Systematic innovation: an introduction to the theory of inventive problem solving (TRIZ)*, *CRC Press*, 1998
30. Varki, E., Merchant, A., Xu, J. and Qiu, X., "Issues and challenges in the performance analysis of real disk arrays," *IEEE Transactions on Parallel and Distributed Systems*, Vol.15, no. 6, 2004, pp. 559-574
31. Wands, B., Digital Creativity: Techniques for Digital Media and the Internet, John Wiley & Sons, Inc, 1997
32. Wikipedia, "RAID," http://en.wikipedia.org/wiki/RAID
33. Wikipedia, "TRIZ," http://en.wikipedia.org/wiki/TRIZ
34. Zhang, J., Chai, K. H. and Tan, K. C., "40 inventive principles with applications in service operations management," *The TRIZ Journal*, 2003
35. 張世彗, *創造力：理論、技法與教學*, *五南*, 2007