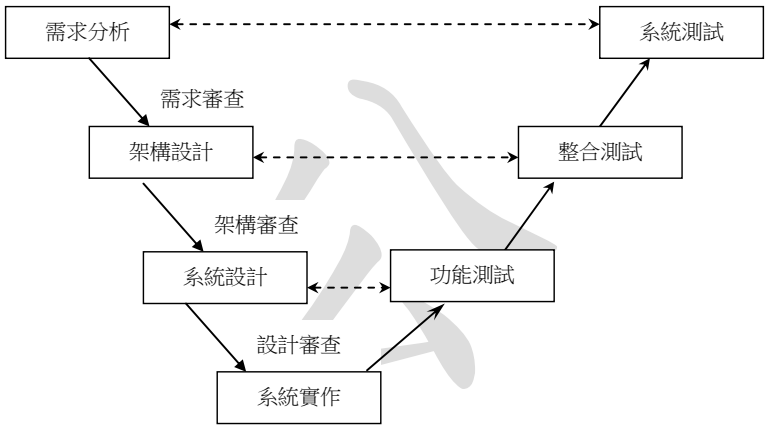
104 年公務人員高等考試三級考試試題

類科：資訊處理
科目：系統專案管理

一、V 模型（The V-Model）為常見的系統開發模型之一，請繪圖並說明其特性，並從系統分析師的角度來探討其優、缺點。

【擬答】：



㈠V 模型是一種 IT 開發方式，是台灣一般大型軟體專案最常用的開發流程，有下列特性：

1. 由瀑布模式改良，並加入確認與驗證（V&V）的基本觀念，展現了不同抽象層次觀點的開發作業與相關的測試作業之間的關係。
2. 一般將軟體專案開發分為三個開發層次，高階的抽象層次處理使用者的需求分析；中階的抽象層次將重點置於將所了解的需求轉化為軟體架構；低階的抽象層次主要為系統功能的細部設計與系統實作。

㈡採用 V-Model 的優點：

1. 配置了良好的結構方法，讓每個開發作業都可以根據前一個開發作業的詳盡產出來進行作業。
2. 測試計劃可以很好地提早在編程前就開始進行規劃，如此將為專案省下大量的時間。

㈢採用 V-Model 的缺點：

1. 無法避免在專案後期需求與設計的經常變動，而影響程式碼品質的問題。
2. 愈高階層次的開發產出，其瑕疵是愈晚才會被驗證出來。相較於低階層次的開發產出，其衝擊影響力又是相當嚴重的，因為這代表瑕疵將會在較低層次的開發產出中擴散與蔓延。

二、目前國內外通常採用軟體能力成熟度模式整合（Capability Maturity Model Integration，以下簡稱 CMMI）或是 ISO 9000 以為企業本身產品（或軟體）開發能力評估與品管標準。而六個標準差（Six Sigma）則是目前工業界盛行的一種品管檢測方式，請說明 CMMI 與 ISO 9000 之異同點。另請探討並繪圖說明 CMMI 階段式表述（Staged Representation）與六個標準差之間的關係。

【擬答】：

㈠CMMI 與 ISO-9000 的異同：

1. 相同點：
   ⑴兩者均強調流程導向。
   ⑵均強調持續改善。
   ⑶對於執行 ISO-9001 的軟體業者而言，多數的流程領域中已有部份流程實行。
2. 差異點：
   ⑴CMMI 對於每一個流程有詳盡明確的要求。
   ⑵CMMI 有階段式提升的規劃。

⑶CMMI 每一階段只有集中焦點在少數幾個流程。

⑷CMMI 強調落實與產生績效。

㈡*1.*如下圖所示，CMMI 階段式表述方法把 CMMI 中的過程區域分成了 5 個成熟度級別，每達成一級成熟度，即代表組織流程能力的增進。因此成熟度可提供在某些特定的專業領域下，預測組織未來績效表現的方法，各包含許多流程領域。經驗顯示，在組織改善的過程中，流程領域的複雜性會不斷增加，若組織能專注於一組可掌握的流程領域，將會有最佳的績效表現。

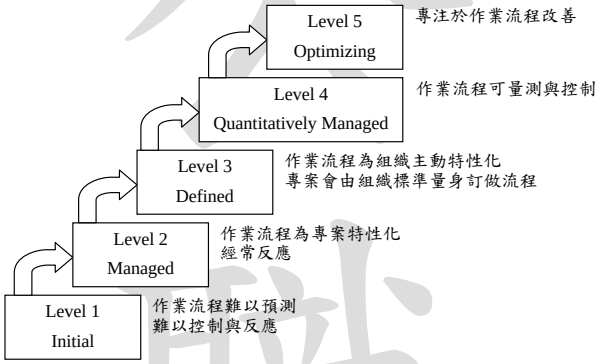
⑴CMM-Level 1（Initial）：流程漫無章法無法預測，控制很差，也缺乏反映。

⑵CMM-Level 2（Managed）：流程已經為專案而特性化，同時經常會有回應。

⑶CMM-Level 3（Defined）：流程已經為組織而特性化，反應積極，專案則從組織標準而調適。

⑷CMM-Level 4（Quantitatively managed）：流程可度量與控制。

⑸CMM-Level 5（Optimizing）：專注於流程的改善。



*2.*6 個標準差的 DMAIC 五個階段步驟，指的是針對企業所遇到的問題進行下列五個步驟：’

⑴界定（Define）D 定義問題，客戶需求和項目目標等等。

⑵衡量（Measure）M 測量當前流程的關鍵方面，收集相關資料。

⑶分析（Analyze）A 分析數據，尋求和檢驗原因和效果之間的關係，確定是什麼關係，然後確保考慮到所有因素。通過調查，發現因為殘疵的根本原因。

⑷改進（Improve）I 提升優化當前流程，根據分析數據，運用不同方法，例如實驗設計、防誤防錯或錯誤校對，利用標準工作創建一個新的、未來的理想流程，建立規範運作流程能力。

⑸控制（Control）C 控制改變未來流程，確保任何偏離目標的誤差都可以改正。執行控制系統，例如統計流程控制，生產板、可見工作區和流程持續改善等。

*3.*因此可見 DMAIC 可用於 CMMI 中各流程領域持續改善中的改進步驟。

三、在資訊系統開發過程中，專案管理者常會因應客戶端的要求而被迫縮短開發時間而將軟體提前釋放，而此種狀況為開發人員及專案管理者所經常面臨到的嚴峻問題與挑戰。根據國內／外研究報告指出資訊系統開發時程的壓縮有其極限性，事實上管理者通常無法任意藉由增加開發人員與添購更多的軟、硬體設備來達到時程壓縮的目的。Putnam 提出了軟體方程式（Software Equation），其定義為：$E=[LOC\times B^{0.333}/P]^3\times(1/t^4)$，其中 E 為開發心力（Development Effort，單位為人月或人年）、t 為專案執行時間、B 為特別技能因子、P 為生產力參數、LOC 為軟體大小（單位為程式碼行數）。請透過軟體方程式來舉例說明開發時程壓縮，對將開發心力造成何種程度的影響。另從實務面來看，合理且可行的時程壓縮極限應為多少？請敘述其可能原因為何？

【擬答】：

㈠根據軟體方程式，開發心力(E) 與專案執行時間(t) 的四次方成反比，也就是說開發時間若縮短一半，則開發心力將成為原來的 16 倍，如此顯然將導致過高的成本，且明顯導致極高的

管理壓力，而無法因應各種風險。

㈡ 根據 Norden 的專案時程回復極限函數(Recovery boundary function) 可知，至專案完工所剩餘時間越短，可回復的落後進度越少。在 COCOMO 模式中，$D = c_b(E)^{d_b}$，E 是用「人月」來計算的工作量(成本），D 是指累積的開發時間（人月），c_b和 d_b則需考量專案類型，因此合理且可行的時程壓縮極限應為到原工作時程的 75%左右。其原因如下：

1. 壓縮的工作應有助於縮短整體的工期：關鍵路徑上的工作項目無法縮短工期時，便無法縮短時程。避免某一工作的時程縮短，卻造成其它工作的時程延長。壓縮後不會影響合理的先後關係。部份壓縮時程的方法將增加成本的支出，長時間工作可能造成效率的低落。
2. 不同開發階段的壓縮成效：不同開發階段的壓縮成效有顯著的差異，分析、編碼階段的時程縮短效果高於整合測試階段。編碼階段的壓縮成本大於分析階段，分析階段壓縮時程對專案品質的影響明顯小於編碼與整合測試階段的影響。
3. 壓縮成效：專案延遲愈少、困難度愈低，資源愈充裕、規模愈大、目標愈明確，其時程壓縮成效也愈好。如果兼顧成本與品質因素，則激勵為最有效的時程壓縮策略。

四、針對下列八支程式模組：

㈠請完成下列表格並明確指出這些程式各具有何種內聚力（Cohesion）及說明其原因？在此內聚力型態（Cohesion Types）須從最差（Worst）至最佳（Best）依序正確排列。另說明欄中若無任何具體說明或解釋逕以零分計算。

內聚力型態	所對應之程式模組（請以 P1,P2,…等標示）	說明
	⋮	

㈡請針對該表格中最差（即 Worst）內聚力型態之程式模組提出具體改進方法。

㈢假設吾人定義內聚力比率（Cohesion Ratio）公式如下，請據此計算出該批程式模組之內聚力比率。

$$\text{Cohesion Ratio} = \frac{\text{Number of program modules having functional cohesion}}{\text{Total number of program modules}}$$

```
//P1
public class P1 {

    public void count1(int m, int n, int p)
    {
        int counter1, counter2, counter3;
        counter1 = 1;
        cusum = 0;
        while (counter1 <= m)
        {
            cusum += counter1;
            counter1 += 1;
        }
        counter2 = 1;
        product = 1;
        while (counter2 <= n)
        {
            product *= counter2;
            counter2 += 1;
        }
        counter3 = 1;
        sum = 0;
        while (counter3 <= p)
        {
            sum += counter3;
            counter3 += 1;
        }
        mean = sum / p;
    }
    public int getSum()
    {
        return sum;
    }
    public int getProduct()
    {
        return product;
    }
    public int getCusum()
    {
        return cusum;
    }
    public int getMean()
    {
        return mean;
    }
    private int sum, product, cusum, mean;
}
```

```
//P2
public class P2 {

    public void count2(int n)
    {
        int counter;
        counter = 1;
        cusum = 0;
        product = 1;
        while (counter <= n)
        {
            cusum += counter;
            product *= counter;
            counter += 1;
        }
    }
    public int getCusum()
    {
        return cusum;
    }
    public int getProduct()
    {
        return product;
    }
    private int cusum, product;
}
```

```
//P3
public class P3 {

    public void count3(int n)
    {
        int counter;
        counter = 1;
        cusum = 0;
        while (counter <= n)
        {
            cusum += counter;
            counter += 1;
        }

        mean = cusum / n;
    }
    public int getCusum()
    {
        return cusum;
    }
    public int getMean()
    {
        return mean;
    }
    private int cusum, mean;
}
```

```
//P4
public class P4 {

    public void count4(int n)
    {
        int counter;
        counter = 1;
        cusum = 0;
        while (counter <= n)
        {
            cusum += counter;
            counter += 1;
        }
    }
    public int getCusum()
    {
        return cusum;
    }
    private int cusum;
}
```

```
//P5
public class P5 {

    public void count5(int first,int second)
    {
        int intermediate;
        intermediate = first;
        result_first = second;
        result_second = intermediate;
    }

    public int getResult_first()
    {
        return result_first;
    }
    public int getResult_second()
    {
        return result_second;
    }
    private int result_first, result_second;
}
```

```
//P6
public class P6 {

    public void count6(int n,int product)
    {
        int counter1, counter2, counter3, counter4;

        counter1 = 1;
        int a[] = new int[n];
        while (counter1 <= n)
        {
            a[counter1-1] = counter1;
            counter1 += 1;
        }
        counter2 = 0;
        cusum = 0;
        while (counter2 < n)
        {
            cusum += a[counter2];
            counter2 += 1;
        }
        counter3 = 0;
        prod = 1;
        while (counter3 < n)
        {
            prod = prod* product * a[counter3];
            counter3 += 1;
        }
        counter4 = 0;
        sum = 0;
        while (counter4 < n)
        {
            sum += a[counter4];
            counter4 += 1;
        }
        mean = sum / n;
    }
    public int getCusum()
    {
        return cusum;
    }
    public int getProd()
    {
        return prod;
    }
    public int getSum()
    {
        return sum;
    }
    public int getMean()
    {
        return mean;
    }
    private int cusum, prod, sum, mean;
}
```

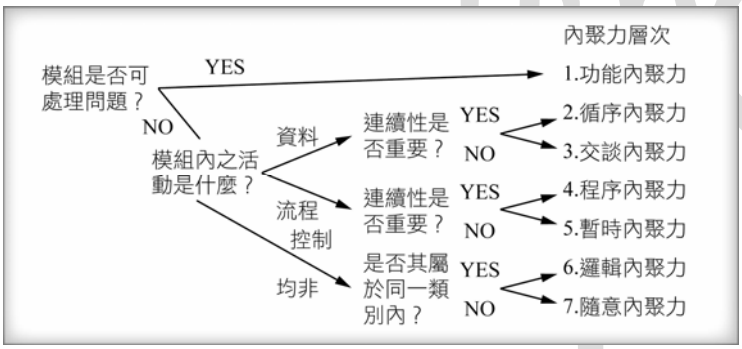
```
//P7
public class P7 {

    public void count7(int[] tmp, int n)
    {
        int counter1, counter2, temp;
        counter1 = 0;
        a = tmp;
        System.out.print("\n");
        for (counter1 = 1; counter1 < n; counter1++)
        {
            for (counter2= 0; counter2 < counter1; counter2++)
            {
                if (a[counter1] <a[counter2])
                {
                    temp = a[counter1];
                    a[counter1] = a[counter2];
                    a[counter2] = temp;
                }
            }
        }
    }
    public int[] geta()
    {
        return a;
    }
    private int[] a;
}
```

```
//P8
public class P8 {

    public void count8(int m, int n, int p, int flag)
    {
        int counter1, counter2, counter3;
        cusum = 0;
        product = 1;
        sum = 0;
        mean = 0;
        if (flag == 1)
        {
            counter1 = 1;
            cusum = 0;
            while (counter1 <= m)
            {
                cusum +=  counter1;
                counter1 += 1;
            }
        }
        else if (flag == 2)
        {
            counter2 = 1;
            product = 1;
            while (counter2 <= n)
            {
                product *= counter2;
                counter2 += 1;
            }

        }
        else
        {
            counter3 = 1;
            sum = 0;
            while (counter3 <= p)
            {
                sum += counter3;
                counter3 += 1;
            }
        }
        mean = sum / p;
    }
    public int getCusum()
    {
        return cusum;
    }
    public int getProduct()
    {
        return product;
    }
    public int getSum()
    {
        return sum;
    }
    public int getMean()
    {
        return mean;
    }
    private int cusum, product, sum, mean;
}
```

［104 高］

【擬答】：

㈠以下列程序判斷內聚力層次



內聚力型態	對應程式模組	說明
功能內聚力		
循序內聚力	P7	各function對相同資料且geta需要接續在Count7後執行
交談內聚力	P2 P3	Count2模組中各function對相同資料但順序不重要 Count3模組中各function對相同資料但順序不重要
程序內聚力	P4 P5	各function對不同資料且getCusum需要接續在Count4後執行
暫時內聚力	P6	各功能對不同資料且連續性不重要
邏輯內聚力	P8	count8中的執行由上層掌握的控制參數flag決定
隨意內聚力	P1	count1模組中的4個功能分別計算cusum, product, sum, mean，基本上其間無關

㈡P1 最差，應該調整模組功能分割，將 count1 中的 cusum 計算功能與 getcusum 合併另外成立模組；計算 product 與 getproduct 合併另外成立模組；計算 sum, mean 與 getsum、getmean 合併另外成立模組。

㈢1.P1 中共有 5 個程式模組，其中有功能內聚力的有 4 個，故 Cohesion Ratio=4/5=0.8。
2.P2 中共有 3 個程式模組，其中有功能內聚力的有 2 個，故 Cohesion Ratio=2/3=0.67。
3. P3 中共有 3 個程式模組，其中有功能內聚力的有 2 個，故 Cohesion Ratio=2/3=0.67。
4. P4 中共有 2 個程式模組，其中有功能內聚力的有 2 個，故 Cohesion Ratio=2/2=1。
5. P5 中共有 3 個程式模組，其中有功能內聚力的有 2 個，故 Cohesion Ratio=2/3=0.67。
6. P6 中共有 5 個程式模組，其中有功能內聚力的有 4 個，故 Cohesion Ratio=4/5=0.8。
7. P7 中共有 2 個程式模組，其中有功能內聚力的有 2 個，故 Cohesion Ratio=2/2=1。
8. P8 中共有 5 個程式模組，其中有功能內聚力的有 4 個，故 Cohesion Ratio=4/5=0.8。