**題目：應用設計樣式發展統整API於多平台社交機器人之開發**

1. 研究計畫內容：

(一)摘要

科技日新月異，人人手上都有一台智慧型手機，花在社交軟體上的時間也日益漸增。現今的商家、企業經常在社交平台上建立會自動回應的機器人向使用者宣傳等等，機器人不只增進回覆客戶的速度又能利用聊天室效果拉近跟使用者的距離，越來越多人嘗試創建自己的機器人。

本計畫預計研發一套統整API，讓開發者開發出跨平台機器人，可以有效降低成本，提升效率。各種社交平台都有不同的API，要新增或修改功能時，必須配合每個社交平台上的規範，透過本計畫的統整API，只需要修改一次，就能透過轉換器將全部社交平台上的機器人的功能修改完成。

本計畫採用軟體工程的觀點，運用設計樣式開發API。設計樣式是為解決軟體設計中特定且常見問題而提出的方案，使程式的耦合降低，較能適應變化。本計畫將使用到數種設計樣式，如Adapter Pattern、Factory Method Pattern、Abstract Factory Pattern以及Template Method Pattern達成統合以及轉換的功能，希望可以讓此API的設計有更好的軟體品質。

(二)研究動機與研究問題

**研究動機**

近年來在眾多調查中顯示人們在通訊應用上的時間遠遠超過使用其它應用程式的時間，而在通訊平台上的應用程式呈現方式為 Bot 。Bot是指可以透過聲音或文字與使用者進行對話的電腦應用程式。許多通訊軟體都釋出機器人API，讓開發人員建立自動回覆使用者問題的機器人，傳送新聞、路況等即時訊息，讓使用者可以更直覺與機器人互動。

本計畫認為機器人的發展性無限，因此構想了一個整合API，可以讓開發者創建出跨平台的機器人。

**研究問題**

每一個通訊軟體都有各自的機器人API，若開發者想自行開發機器人，就必須使用該通訊軟體所釋出的API。開發前必須了解API的設定，例如FB需要先建立粉絲專業才能進行後續步驟，以及明白API運作方式，所以當開發人員想同時建立多個社交平台機器人時，會有許多操作上的疑慮，本計畫希望能統一API並提供給開發者，降低開發者的開發成本。

(三)文獻回顧與探討

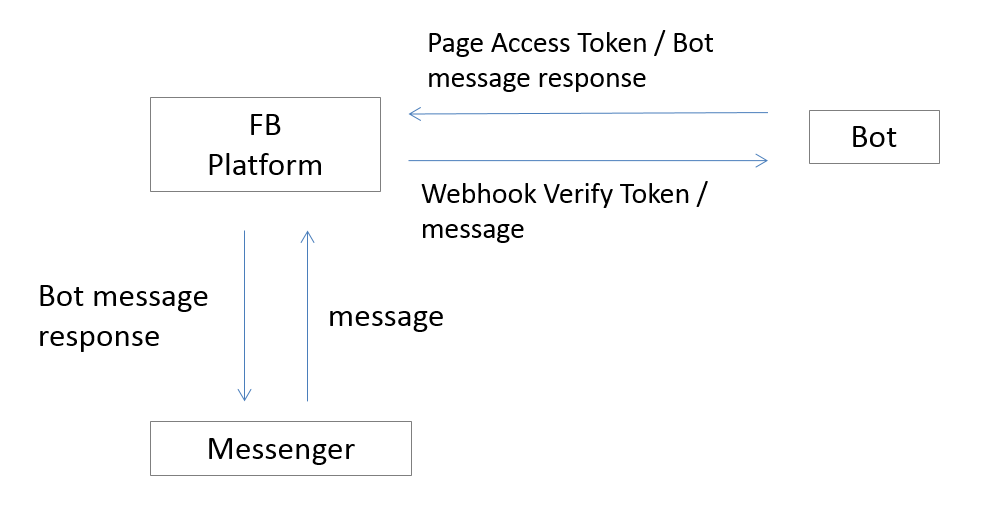
API，全名Application Programming Interface，是一種抽象的概念，僅定義一個介面，不涉及應用程式在實作過程中的具體操作，通常是特定軟體、系統等等。開發者為了方便使用者在該平台下進行衍生創作或是開發，而提供類似規範的管道，循著提供的規則走，可使用到更客製化的功能。

為了瞭解社群機器人API的運作方式，本計畫預計採用Facebook [1]、LINE[2]、PLURK[3]進行說明如下：

1. Facebook

Facebook Messenger Bot運作方式如圖一所示。開發者在Facebook申請粉絲專頁後，可以取得存取權杖，由權杖確認Bot是否能與Facebook聯繫。Facebook平台依照在Webhook設定的網址( URL) 做出回呼，傳回「200 OK」HTTP 回應，代表該網頁被成功傳回，經過上述的平台與開發者建立的Bot驗證後，當使用者透過Messenger傳遞訊息時就能接收到由Bot所做出的回應。

使用者透過Messenger傳遞訊息，經由Facebook平台的傳送，當Bot接受到訊息請求時，以POST的方式回應。Facebook平台與Bot之間是以固定的訊息結構溝通。在結構上涵蓋recipient 物件以及message 物件，在recipient 物件中描述接收者的編號；在message 物件中描述訊息型態。



圖一 Facebook Messenger Bot運作方式

1. LINE

LineBot 的運作模式與Facebook相似，如圖二所示。開發者向 LINE Business Center 申請帳號，透過 Line Developer 模式申請 Bot API 取得 Channel ID、Channel Secrect 和 Channel Access Token ，當使用者透過 Line 傳遞訊息時，HTTP POST request 會透過平台設定的Webhook 網址( URL )提出要求確認執行環境可與 Bot取得聯繫，不論Webhook、Reply message或 Push message，皆會有Request headers和Request body。Request headers包含X-Line-Request-Id，為數位憑證 ( Signature validation )的數值，用來辨識此要求來自LINE平台，Request body則以陣列的方式用來描述事件物件 ( event objects )，其包括replyToken、type ( 是否為message )、timestamp ( 時間 )、source ( type、userId )、message ( id、type、text )，其中source的type是指使用者傳送訊息的來源是個人或群組；message的type是指訊息傳送的形式 ( 文字、圖片、貼圖、網址……等 )，待伺服器端回傳「200 OK」Status code，代表該網頁被成功建立，訊息也能回傳。



圖二 LINE Bot運作方式

1. PLURK

PLURK的運作模式如圖三所示。開發者首先要註冊一個新帳號，到PlurkApp的「註冊新的應用服務」中選擇機器人。填寫相關資料，就能得到Plurk提供的App Key和App Secret，再經過OAuth 的授權驗證，會再得到Access Token及Token Secret，Access 官方提供的library裡，有程式碼的參數需要置換成上述四個亂碼。將主程式碼與library放在同資料夾中，當程式碼運作，會透過那四個授權碼跟平台進行驗證，通過後就能運行機器人。

PLURK的message有一定的物件結構，其中的格式會包含plurk\_id、qualifier、limited\_to等等，分別會描述使用者唯一的id、發送訊息時的狀態、限制有誰能看到此訊息等等。結構與上述兩項社交平台差異在於PLURK機器人並沒有限制於一對一，因此它的物件有一定的數量且缺一不可。

Untitled Diagram.png

圖三PLURK Bot運作方式

本計畫欲在不更動原社群軟體所釋出 API的情形下完成API統整，達到跨平台的目的，因此會使用到Adapter Pattern、Factory Pattern等等的Design Pattern。

Design Pattern為解決特定且常見問題的方法，使得程式可以具有較高的重用性與彈性，讓往後的維護或擴展更加方便。以下將從兩個層面探討Design Pattern。

1. Design Pattern的結構

每個設計樣式均有結構性描述，涵蓋下列四項說明：

1. 樣式名稱：以簡易名稱表示樣式，方便使用者往後的使用、溝通或是文件說明。
2. 問題描述：描述樣式所解決的問題與問題發生情境。
3. 解決方式：描述設計樣式的組成、技巧和元素之間的關聯、責任與合作。
4. 影響：描述使用該樣式的結果與優缺點。
5. Design Pattern的分類

由用途與範圍來進行分類：

1. 可由用途分成三種：
2. 創建：控制物件的生成。
3. 結構：了解元件之間的關係。
4. 行為：考量物件間的互動、交流。
5. 從範圍可分成兩種：
6. 類別：主要以繼承的概念來解決問題。
7. 物件：主要以組合的概念來解決問題。

總共會有6種不同的組合：

生成配上類別範圍時，物件的生成有部分會延遲到子類別時再決定；配上物件範圍時，則是將部分物件生成委託給其他物件生成。

結構配上類別範圍時，實作與組合介面皆使用繼承；配上物件範圍時，用物件組合來達成彈性和新的功能。

行為配上類別範圍時，用繼承來分配類別間的行為；配上物件範圍時，物件之間的組合並不是繼承，是對等物件協力完成工作。

在 [4]、[5] 中均有提及詳細設計樣式，下面將簡單介紹本計畫欲使用的設計樣式。

1. Adapter Pattern

轉接器模式，分為 Class Adapters 和 Object Adapters。 Class Adapters 是指在存在一類別的情況下，此類別稱為 Adaptee 類別，其功能與客戶端需求相符，但與客戶端所期待的介面有所出入，以不更動舊有 Adaptee 類別或無法取得其程式碼為前提，新增一個類別透過 Adapter 轉接，使 Adapter 可以繼承既有功能 ( Adaptee 類別 )並達到客戶需求，如圖四所示； Object Adapters 則是以合成為主，主要特性為一個 Adaptee可配合多個類別，其所有的子類別都能動態抽換，缺點是單一 Adaptee 的程式碼會太過龐大。如圖五所示。

Adapter2.jpg

圖四 Class Adapter 運作模式

Object2.jpg

圖五 Object Adapter 運作模式

1. Factory Method Pattern

工廠方法模式，定義建立物件的介面，由次類別決定要實體化的類別。是將 new Class封裝成一個 Factory Class來負責實體化類別，修改Factory Class時，不會影響到使用者操作，還可隱藏物件生成的過程，如圖六所示。

Factory.png

圖六Factory Method 運作模式

1. Abstract Factory Pattern

抽象工廠模式，沒有明確定義實體類別的情況下，提供介面建立出相關物件。將一組具有關連或相依性的物件，利用同一介面物件產生。缺點是一旦要擴充產物，就必須改變每個次類別的介面。Abstract Factory 經常是由多個 Factory Method 組成。如圖 (7) 所示。

Abstract (1).png

圖七 Abstract Factory 運作模式

1. Template Method Pattern

樣板方法模式，將一個演算法骨架定義好，演算法裡會用到的方法或步驟，在不改變演算法架構下，用次類別去定義。兩個抽象方法，其內部步驟有相似的部分，抽出相似部分作為定義，剩下的部分用次類別去填補。在圖八中TemplateMethod具有operation1( )、operation2( ) 兩個方法，其中operation2( ) 為抽象，交由ConcreteClass給予具體定義。

templateMethod.png

圖八Template Method運作模式

如何套用上述設計樣式達到API整合，將在(四)研究方法及步驟做更詳細的說明。

(四)研究方法及步驟

圖九為本計畫的架構圖。

Structure (1).png

圖九 架構圖

Bot Development Toolkit：提供給開發者使用，開發出跨平台機器人。

Bot Uniform API：統整的API，本計畫的目的。

Bot Adapters：將Uniform API轉換成各社交平台上的API。

Social messager Platform：社交平台、軟體等等。

Developers' Bot：開發者運用本計畫預計做出的整合API開發出的機器人。

使用者對開發者開發的機器人傳訊，會經過社交平台上釋出的API傳給機器人，而機器人回傳的訊息會先經過Uniform API再至對應的Adapter，轉換成符合使用者所在的社交平台上釋出的API，最後訊息傳給使用者。

本計畫有些許部分會使用到物件導向[6]的概念，且預計使用的語言為JavaScript[7][8]，並運用Node.js[9][10][11]此執行環境。

JavaScript是一種直譯式語言，不需編成機器碼即可執行。直譯式語言開發效率高，省略編譯程式的麻煩，且只要直譯器可在不同平台運行，同一套JavaScript的程式碼也可以跨不同的平台，不需重新開發。

Node.js是一個JavaScript的開發平台，除了是OpenSource外還可跨平台。當Node.js接收到一筆使用者連線時，會請OS將連線放入heap中保留，讓連線進入休眠狀態，直到系統通知才觸發callback。此處理方式的優點是不會用到CPU資源，只占用記憶體，且因為JavaScript的特性，每個要求都會有一個Callback，可避免Block發生。

預期依下列步驟進行：

1. 分析API運行方式：

本計畫的實驗主體預計採用Facebook、LINE、PLURK來實行跨平台運作，為了瞭解各API運行的方式，實際操作其釋出的API，將先針對Facebook、LINE、PLURK的訊息結構進行分析。

1. Facebook對於結構描述包含recipient 物件以及message 物件，前者描述接收者的編號，後者則是描述訊息型態、內容。
2. Line的結構主要包括 source 和message，前者分辨訊息來源為個人或群組編號，後者接收發送者編號、訊息型態、內容。
3. PLURK的結構格式會包含plurk\_id、qualifier、limited\_to等等，分別代表使用者唯一的id、發送訊息時的狀態、限制有誰能看到此訊息等等。

三種訊息結構相同的部分有訊息內容、接收者的ID等等。接著根據以上的分析結果，先行製作出會重複使用者說話的機器人達到分析API的目的。

1. 抽離出三社交平台API相似或相同的部分並統一訊息整體架構：

根據整理的資訊，將API相同部分、相似部分以及不相同的部分統整出來。例如訊息傳遞中，皆會包含送訊息者的ID或內容等等，本計畫設想統一訊息結構，在此結構中欲涵蓋接受者編號、傳送者編號、訊息型態與訊息內容。而其他將分類為不相同的部分，將在下一步驟運用設計樣式進行處理與說明。

1. 探討設計樣式套用在統整API上的設計：

依循步驟二已抽離統一的架構，再由Abstract Factory Pattern分析屬於哪一個產生物件的類別，此時是利用Abstract Factory Pattern包含的Factory Pattern將訊息處理完成，例如依照訊息的型態，分成文字、表情貼圖……等等，若為文字就會到開發者的資料庫搜尋關鍵字，貼圖的判斷則由分析代碼作出相對應的回應。最後經由Adapter轉換成該社交平台的結構物件，再用Template Method的概念，將各社交平台結構物件中不相同的部分加上去。

1. 實作並運用此整合API開發的機器人：

本計畫預計做出一個生活提醒機器人，此機器人可以根據使用者的要求，在特定時間提醒使用者，用此機器人去驗證此API能確實運作。先確認該機器人能夠接收到來自不同平台的訊息，也可以發訊息給不同平台上的使用者，且測試當透過整合API修改機器人功能時，能成功同時反映到在三個社交平台上的機器人運作。

(五)預期結果

1. 實際做出生活提醒機器人，驗證可以達到跨平台的目的。
2. API以Open Source的方式提供給開發者，其可自行開發跨平台機器人。

(六)參考文獻

[1] https://developers.facebook.com/docs/messenger-platform

[2] https://devdocs.line.me/en/

[3] http://www.plurk.com/API

[4] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides ( 1994 ). *Design Patterns: Elements of Reusable Object-Oriented Software.* Reading, Mass.: Addison Wesley

[5] Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra ( 2004 ). *Head First Design Patterns.* Sebastopol, CA: O'Reilly

[6] Brett McLaughlin、Gary Pollice、David West ( 2007 ) 深入淺出物件導向分析與設計(楊仁和，譯)台北市：歐萊禮(原著初版出版於2006年)

[7] Eric T. Freeman、 Elisabeth Robson ( 2015 ) 深入淺出 JavaScript 程式設計(蔣大偉，譯)台北市：歐萊禮(原著初版出版於2014年)

[8] Ethan Brown(2016)JavaScript 學習手冊(第三版)(賴屹民，譯)台北市：歐萊禮(原著第三版出版於2016年)

[9] 錢逢祥、蔡政崇、林政毅 ( 2014 )。 不一樣的Node.js：用JavaScript打造高效能的前後台網頁程式。臺北市：松崗資產管理。

[10] Shelley Powers( 2016 ).Learning Node, 2nd Edition.O'Reilly Media

[11] Mario Casciaro ( 2016 )Node.js設計模式(柳百郁，譯)新北市：博碩文化(原著初版出版於2014年)。

(七)需要指導教授指導內容

1. 每周進度審查與建議
2. 研究方向指引
3. 實作系統規畫方向
4. 書籍、論文推薦及導讀