軟體工程期中報告

Software Engineering Midterm Report

以UML分析德州撲克專案

Analysis of Texas Poker Project by UML

指導老師：吳宗勳講師

組員：陳柏宇、梁日瑋、林煌錡、姜冠芸

中華民國104年10月28日

1. **摘要**

這個德州撲克專案(Texas Poker Project，以下簡稱為TPP)，是103學年時在機率與統計課程上，由陸儀斌老師所指導，學生分為六組各別實作而成的專案。六組裡其中五組負責客戶端(Client)介面的設計，進行玩家的動作與人工智慧決策；最後一組負責伺服端(Server)的開發，完成德州撲克遊戲流程的設計、與網路連線通訊的規劃。

TPP Server的開發是使用Golang所撰寫而成，是一套極輕量化的Server。雖然其中的設計在規劃時有粗略地加入物件導向的概念，但是在第一版(TPP 1.0)完成時，主體的程式碼長達2000多行，僅僅能提供運作而已，無論是修改亦或是交接都有著巨大的困難。於是，為了交接與維護，後來TPP進行了一次大規模的程式碼重整(Refactoring)。將主體的程式碼盡可能地除耦合，按照功能與關連分別置入不同的Package與Class，便有了現在的TPP 2.0。

然而，按照嚴格的物件導向分析與設計標準而言，這套系統仍然是漏洞百出。因此，趁著本次軟體工程課程的期中報告，我們選擇這個系統作為主體，重新再以物件導向的概念檢視它，並實際地以UML繪出圖表，希望能為下一次的改版奠定基石。

1. **德州撲克規則簡介**
2. 輪到玩家行動時，玩家可選擇的行動包括：
3. 蓋牌：放棄手中的牌，不參與該局。蓋牌後，玩家便無法贏得底池獎金，也不可取回投入底池的籌碼，包括盲注。
4. 跟注：指下注的金額與前一位玩家的下注金額相同，表示玩家繼續餐與該局。請注意，在翻牌圈之後，如果之前沒有玩家下注，當局玩家可以選擇過牌。
5. 加注：指下注的金額較前一位玩家的下注金額高。如果之前沒有玩家加注，當局玩家可以自己加注。
6. 接下來介紹各個階段：
7. 下盲注：在底牌(即各個玩家的兩張手牌)發下前，強制莊家的下一位玩家下「小盲注」，再下一位玩家再下「大盲注」(通常為小盲注的兩倍)，接下來便進入下一個階段。
8. 翻牌前(首輪下注)：此時各個玩家發下兩張手牌(只有玩家自己知道)，玩家看完自己手牌後，並由「大盲注」的下一位玩家開始下注，可以做的動作即蓋牌、跟注及加注，直到莊家結束後進入下一輪下注。



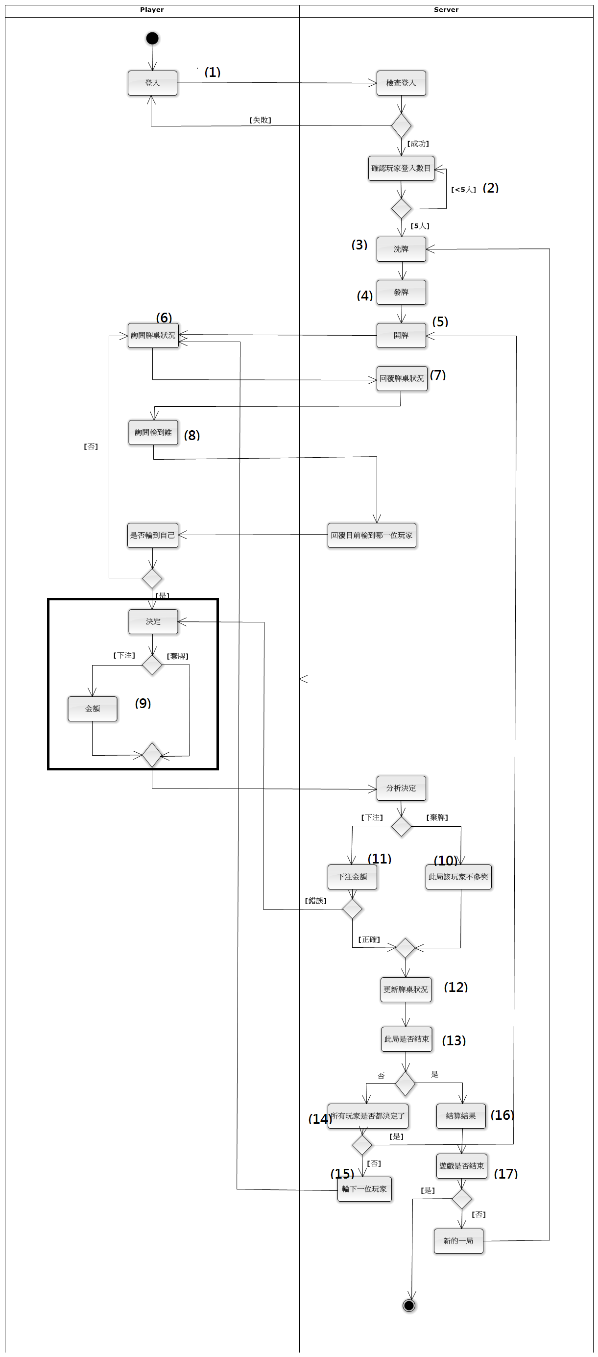
1. 翻牌圈(第二輪下注)：首輪下注結束後，會發下3張公用牌，稱為「翻牌」，然後開始進行第二輪下注，同樣從順時針方向莊家的下一位玩家開始。

****

1. 轉牌圈(第三輪下注)：第二輪下注結束後，會發下第四張公用牌，稱為「轉牌」，然後開始第三輪下注。
2. 河牌圈(第四輪下注)：第三輪下注結束後，會發下第五張公用牌，稱為「河牌」，這是最後一張公用牌。接下來開始第四輪下注。
3. 攤牌：最後一輪下注結束後，尚未蓋牌的玩家必須從2張底牌和5張公用牌中，選出一組5張的最佳牌組，並亮出自己手上的牌，稱為攤牌，此時擁有最佳牌組的玩家獲勝。如果出現平手（兩位以上玩家的牌組一樣好，且同時為最佳牌組），由平手的玩家平分底池。此時牌局結束，下一局開始，莊家按鈕要往順時針方向移一個位置。



1. **活動圖分析**



本系統的流程主要分成玩家以及伺服器端兩大部分。

首先要進行一場遊戲必須先有足夠的玩家，因此由(1)玩家部份發起登入，當(2)登入人數到達五人(預設)時，伺服器端會進行(3)洗牌、(4)發牌、(5)開牌等動作(由於老師要求牌組是每組都看的到的，因此可以先開牌再進行押注)，至此伺服端會開始等待玩家的請求並做出回應。

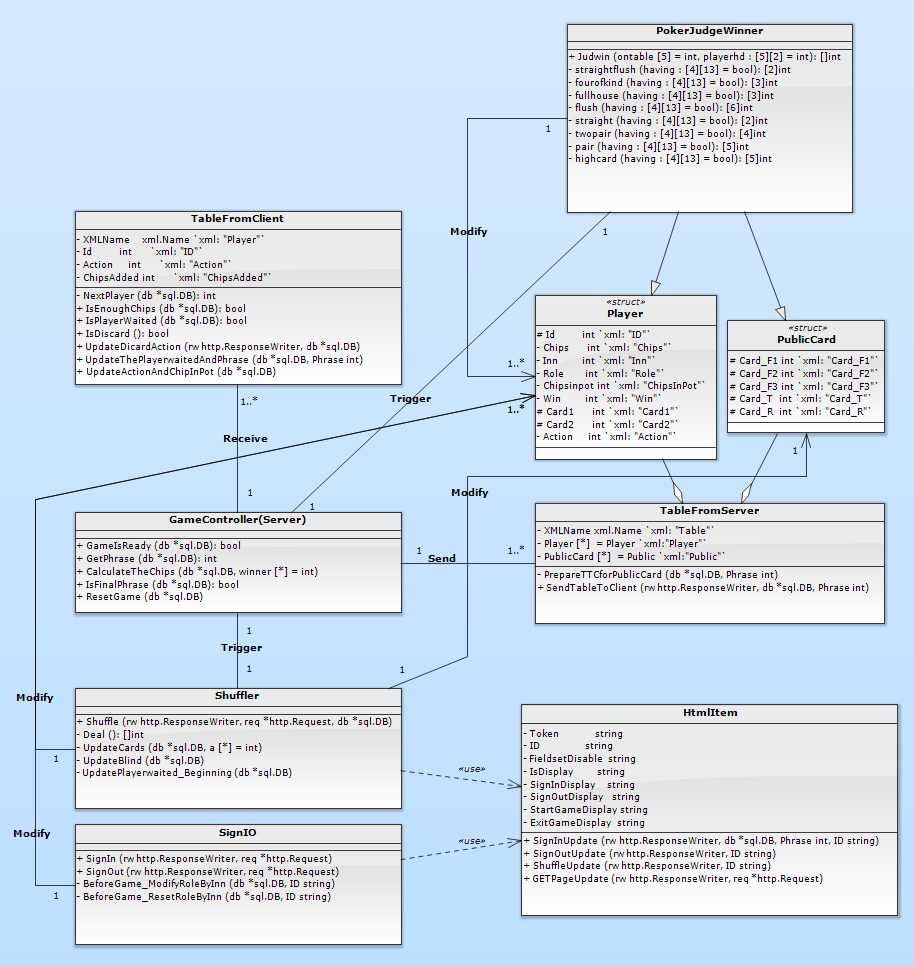
當一個牌圈開始時，(6)每個玩家會不斷向伺服端請求更新牌桌狀況，等到(7)伺服器回應後(8)玩家會詢問是否輪到自己下注，如果伺服器回應尚未輪到玩家，玩家便會從更新牌桌狀況的流程開始，直到輪到自己下注。

(9)當輪到玩家下注時，玩家可以決定自己要棄牌或者要下注多少金額，若(10)玩家棄牌則玩家放棄這一局，(11)若下注則伺服器端會判斷玩家下注金額是否正確(不能低於上一注)，若不正確則重來，正確則(12)更新牌桌狀況，並(13)判斷該局是否結束。

若該局尚未結束則(14)判斷是否該輪所有玩家都已壓完注若尚未完成則(15)輪到下一個玩家押注，若所有玩家皆已壓注則進入下一輪壓注(即從(5)開牌開始)。若該局結束則(16)結算結果並且(17)決定是否繼續遊戲。

1. **類別圖分析**

關於TPP的類別圖，大致可分為七個部分：TableFromClient、TableFromServer、GameController(Server)、HtmlItem、Shuffler、SignIO與PokerJudgeWinner，其中TableFromServer由兩個結構Player與PublicCard聚合而成，整體架構如下：



1. TableFromClient：

此類別意指從客戶端所得來的動作資訊，由於資料傳輸使用XML標準格式，故命名為「TableFromClient」。在這個類別中，屬性包含玩家的識別編號(ID)、動作(Action)與加注籌碼(ChipsAdded)。

而由上述屬性衍伸出的判斷函式有：判斷加注籌碼是否足夠(IsEnoughChips())、是否為輪到的玩家(IsplayerWaited())與是否棄排(IsDiscard())。此外，於資料庫底下，關連到更新(Update)以上三種狀態的函式，也同樣歸屬於此類別。

此類別在架構中僅關聯於GameController(Server)，接收來自客戶端信息後，便直接交由Server進行處理。

1. TableFromServer：

此為GameController(Server)所準備之XML格式資訊，包含著牌桌上所有玩家角色、手牌、籌碼、動作、與目前已翻開之公共牌資訊……等等。此為為了讓玩家瞭解當前牌桌整體狀況所設計之類別。

當玩家向Server發出請求資訊(Info=GetTableSituation)時，Server便會做出處理，觸發TableFromServer內的SendTableToClient()函式(其中會執行一個準備XML資料的封裝函式PrepareTTCforPublicCard())，執行回覆Client請求的動作。

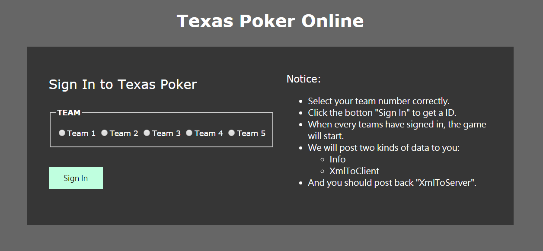
1. GameController(Server) ：

這個部份真的很難定義成一個類別，可是它確確實實地是許多重要函式所集合而成的整體。其中包含判斷遊戲是否完成開始準備(GameIsReady())、取得遊戲階段(GetPhrase())、計算底池籌碼(CalculateTheChips())與重設遊戲(ResetGame())……等等。

隨著遊戲流程的進行，GameController會因應變化而做出不同的反應，也負責著Client與Server資訊的傳輸與處理。

1. HtmlItem：

在開始討論這個物件之前，這邊要先提一下TPP的登入模式。在伺服器架構裡，除了處理遊戲流程之外，還負責著一個登入遊戲頁面。而這個頁面有著幾個元件需要隨著玩家的登入進行控制，例如：登入登出按鈕、遊戲開始訊息、與離開遊戲訊息……等等。

也因此，這個物件負責控制上述元件的顯示與否，也實作了一些相關的資料庫更新函式。直接為Shuffler與SignIO兩個涵式集合所使用，也可以說是依賴(Dependency)。

1. Shuffler：

由於伺服端與客戶端是由分開的小組進行實作(當初實作的目的並非在於撰寫遊戲，而是檢驗各組同學實作機率計算以進行遊戲決策的能力)，因此在遊戲的運行上無法做到完全自動化，在遊戲重新開始的洗牌(Shuffle)動作，是由伺服端這邊掌控以進行新的牌局。

Shuffler行為很單純，經GameController(Server)的控制，Shuffle()使用了HtmlItem的ShuffleUpdat()函式進行相關資料庫的更新，也因此會修改(關聯)到Player與PublicCard的內容。

1. SignIO：

SignIO顧名思義就是Sign In與Sign Out，與Shuffle所處的架構位置相仿。同樣使用HtmlItem的函式，也同樣會進行相關資料庫的更新。而與Player關聯則是因為修改了ID與Role屬性。

1. PokerJudgeWinner：

此類別多重繼承Player與PublicCard，其實也就是根據所有玩家的手牌與桌面上的公共牌來決定誰贏誰輸。未封裝的函式只有判斷輸贏Judwin()，而其他諸如判斷同花、順、對、高牌……等等函式均封裝於內部，不為外部所直接使用。

與Player關聯在於判斷完成後，會賦予(修改)屬性Win值，GameController(Server)以此完成遊戲流程的運作。

1. **小結**

經過了上述的整理與分析，對於TPP的架構我們確實有了更透徹的了解。而同學之間的相互腦力激盪，也產生了許多不錯的新想法，之後在改版時或許可以用上。

雖然在開始之前就已經知道這個系統是很有問題的，但是在仔細檢視與繪出圖表之後，我們才了解到它的不成熟與贏弱。所謂「帶有壞味」的程式碼，應該就是指這種難以維護而充滿耦合性的設計。

我們還不知道要如何才能將TPP給完善到「可以拿出來看」，但是我們相信建置UML圖是開始的第一步，接下來就是看我們如何去改良這些圖，使它整體架構可以真正符合物件導向程式設計的精神。

1. **參考文獻**
2. 趙光正(譯)(民96)。UML精華第三版：增訂嵌入式系統與工作流程(原作者：Martin Fowler）。臺北市：台灣培生教育出版股份有限公司。
3. Google. (2015, October 28). The Go Programming Language. Retrieved from <https://golang.org/>
4. IntelliPoker(民104年10月28日)。德州撲克遊戲規則 – IntelliPoker。取自<https://www.intellipoker.tw/articles/?id=45>