自動議約助理軟體

陳世峰　　苑守慈

francis@im.fju.edu.tw　　yuans@tptsl.seed.net.tw

輔仁大學資訊管理學系企業智慧管理研究室

http://bim.im.fju.edu.tw

摘要

電子商務近年來蓬勃發展，各種研究與應用領域都已吸引許多專家學者投入研究，但在自動議約方面的工作卻因爲牽涉到產品及服務的描述規格標準化，以及協商的支援或自動化能力，因此實作上有其困難性。但根據研究，這方面的工作又是需要的，因此本研究運用一個群體助理軟體的基礎環境，以提供一個能快速建構具有自動議約能力的群體助理軟體的一個解決方案。

關鍵字詞：自動議約，自動化協商，多屬性評估，群體助理軟體基礎環境。

1. 簡介

電子商務近年來蓬勃發展，各種研究與應用領域都吸引許多專家學者投入研究。其中的自動議約(electronic contracting)，乃是一新的熱門研究課題。Baum & Perrit(1991)對自動議約下的定義爲：

"Electronic contracting involves the exchange of messages between buyers and sellers, structured according to a prearranged format so that the contents are machineprocessible and automatically give rise to contractual obligations"

而 Runge(1998) 亦提出：一個自動議約包含了協議之協商(agreement negotiation)和協議之簽訂(agreement signing)兩個程序。協議之協商是根據協商的狀況，在買賣雙方之間做資訊交換的行爲。而協議之簽訂是對協商結果進行簽約的行爲。

綜合以上討論，我們歸納出自動議約有下列幾個重要程序：

1. 以一個通用的溝通機制進行溝通
2. 以自動化(或半自動化)進行協商或合作
3. 協議的簽訂與執行

Runge(1998) 曾提出的一個關於自動議約的調查報告：

"The majority of users In the WWW regard electronic contracting as important and would be willing to negotiate all kinds of contract conditions within the first 15 minutes by themselves."

根據其研究報告指出，消費者希望能在15分鐘內完成自動議約。而依根據消費者購買行爲模型(Consumer Buying Behavior Model)[14]，這段時間內消費者必須完成需求確認(Need Identification)、商品仲介(Production Brokering)、商家仲介(Merchant Brokering)和協商(Negotiation)四項工作。要在很短的時間內，完成上述的每項工作，這說明了一個能協助完成自動議約的系統之重要性。

另外，Dutta & Segev(1999)曾從四個 P(Product，Price，Promotion 和 Placement)和一個 C(Customer Relationship)的角度去探討企業如何在網路市場(marketspace)中順利轉型。其中與自動議約最有直接相關的項目是訂價方式的轉型(transforming of pricing)，包含客製化的價格(dynamic customization of price)、價格的協商(online price negotiation)等工作，目前的達成率只達到 12%，而研究報告亦指出事實上的達成率應會更低。因此我們認爲一個自動議約的環境，將可支援這些問題，加速企業轉型。但在自動議約方面的工作卻因爲牽涉到目的物的描述規格(ontology)標準化，以及協商的支援能力(或自動化能力)，因此實作上有其困難性。但根據上述研究顯示，這方面的工作又是非常需要的，因此本研究希望能提供一個自動議約的解決方案。

在自動議約三個重要程序中，由於協議的簽訂與執行的機制是電子商務的基本環境[22]，而且目前已有許多研究成果，因此本研究只著重於前兩項工作。另外，有鑑於(群體)助理軟體化((multi)agent-based)的電子商務已日漸盛行，因此我們認爲若能以一個通用的助理軟體化的溝通機制，包含溝通協定以及產品和服務的規範，及支援使用者進行協商的機制，將使得群體助理軟體(multiagent system)有共通的基礎進行自動議約。

依據以上的描述我們認爲，要建立一個能支援自動議約的群體助理軟體的基礎環境前，有幾個工作需要完成：

1. 建立一個通用的溝通機制以便助理軟體進行溝通
2. 找出一個能自動化(或半自動化)進行協商的機制

以下即上述工作目前的相關研究成果。

Automated Agent Cooperation

群體助理軟體的成員之間必須透過合作協定進行溝通以達成彼此的協議，使得群體助理軟體能夠共同合作解決問題。而以往的群體助理軟體的設計，將助理軟體的功能與合作協定兩者實作在一起，使得助理軟體的實作變的複雜與困難。這樣的作法有下列幾個缺失：

(1) 合作協定無法重複使用：

助理軟體合作協定的範圍若牽涉到助理軟體的工作層次。則對於每種不同的應用，都會有不同的合作協定，如此合作協定就無法重複使用。而如果合作協定可以重複使用，則不同的助理軟體就可以透過相同的協定來進行合作。同時只需透過一些簡單的反應機制，就可以讓助理軟體合作自動化，這樣的作法就會使助理軟體設計變的更容易迅速、簡單。

(2) 助理軟體很難重複使用：

若助理軟體所使用的合作協定不盡相同，致使不同的助理軟體之間可能無法溝通。這種缺失使得助理軟體重複使用性降低。而如果助理軟體可以重複使用，則助理軟體不需修改，就可以由一個群體助理軟體應用在另一個群體助理軟體中，或甚至應用在多個群體助理軟體

中，降低大量的開發成本。

這樣的作法使得助理軟體的重複使用性不高，而且每個助理軟體都是爲特定的群體助理軟體而實作的，所以很難移植到不同的群體助理軟體。而根據 Wu & Yuan(1999)的研究指出，若將每個助理軟體視爲一個提供某些特定的服務、資訊或應用的基礎實體，並以一個通用界面包裝，則發展人員要實作助理軟體時，只要著重於各個助理軟體的功能面考量，而將其他工作透過那通用界面完成。如此，一個助理軟體完成後，即可直接嵌入於不同的群體助理軟體之中。舉例來說，當某助理軟體可以在某群體助理軟體中扮演零售商的角色，那它也應該能夠在另一群體助理軟體中扮演零售商的角色。由上面所描述的觀點來實作助理軟體，會簡化整體上的設計，並且也使得助理軟體具有重複使用的彈性。

Ontology

在助理軟體與其他助理軟體互動時，對其應用需有一個共同的體認，即爲 ontology。例如助理軟體在買賣汽車時，可能會面臨汽車的許許多多的名稱，如"car"、"automobile"等同義的詞彙，如果沒有汽車的 ontology(包含同義詞，意義等)存在，助理軟體根本無法對進行溝通。如同 Beam & Segev(1997)所言，ontology 的功用在於確保助理軟體確實參照同一物品，而 Princeton University 的 WordNet[26]即是這種定義詞彙意義的基礎 ontology system。

目前已有許多的研究在從事 ontology 的制定，如 Ontolingua[13]和 CommerceNet 的 eCo Framework[12]等。Ontolingua 是利用 KIF(Knowledge Interchange Format)所定義的一種 LISP-based ontology system。University of Edinburgh 曾利用 Ontolingua 定義了一套 Enterprise Ontology[25]，其中描述了企業的組織、活動、任務等等。

eCo Framework 是以 XML 爲基礎的 ontology，透過 XML 的特性，eCo 可以 ontology 的制定容易而且是機器可識別的。Ontology.org 和多家有引響力的公司皆已支持 eCo Framework。因爲只有在一個通用的標準下，助理軟體才能進行各種溝通合作工作。

Multiattribute Negotiation

人對人的協商費時費力，而以助理軟體進行或協助協商，並不會有此問題。同時有研究指出，即使個別助理軟體的行爲簡單，但其所形成的整體環境，仍能以有智慧的方式運作[17]。因此透過賦與助理軟體具有自動化協商的能力，將能以有效率的方式達到使用者的目標。

目前已有一些研究探討自動化的協商機制，最早期的研究成果是 Kasbah[6]。使用者提供想要購買物品的可接受價格(acceptable price)以及時間因素(change over time)等資訊，Kasbah 就依使用者的輸入的資訊進行協商。這是最早的研究，因此有一些缺點存在，例如無策略應用的考慮，只考慮價格因素等等。此後亦有 AuctionBot[2]之類的研究出現，但仍然只考慮價格因素。

Bichler, Kaukal, & Segev(1999)提出一個多重屬性拍賣(multiattribute auction)機制的解決方法，即是以效用函數(utility function)決定提案(proposal)的價值，如此就能以虛擬錢幣(virtual currency)進行交易或協商。此研究最大的關鍵在於效用函數的訂定，它是以決策分析(decision analysis techniques)去決定效用函數，那是一種類似 MAUT(Multiple Attribute Utility Theory)、AHP(Analytic Hierarchy Process)和 conjoint analysis 的技術。它將各屬性賦與一權值(weight)，以決定整體的價值，如下式所示。

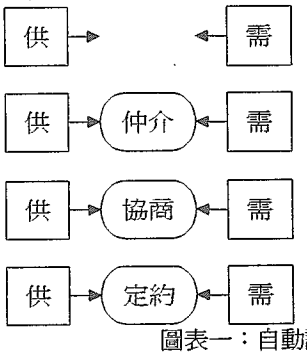
$$U(X_j) = \sum_i^n w_i U_i(X_j^i)$$

以此種作法有兩個缺點。第一，權值的決定無法確實反映使用者需求，因爲要使用者決定出每個屬性的權值並不容易。第二，此種效用函數只考慮到各屬性的線性關係，卻忽略可能存在的非線性關係，仍有其實際運用上的問題。

Oliver(1996)亦提出一具有多重屬性協商(multiattribute negotiation)能力的機制，此機制以基因演算法(genetic algorithm)作爲產生提案的方式，使其具有自動化協商能力。Matos & Sierra(1998)則是以 case base 和 fuzzy rules 作爲產生提案的方法。而 Guttman & Maes(1998a)則是使用 CSP(Constrain Satisfaction Prdblem)的方式處理，但這些研究都是使用 MAUT 的方式處理效用函數，仍然無法去除上述兩個問題。

Electronic Contracting

所謂的自動議約，我們認爲那是以自動化(或半自動化)的方式滿足供(supply)需(demand)雙方需求的行爲(此處所謂的供和需的角色並不限定於單純買賣行爲的雙方)。下圖描述了自動議約在供需之間進行的工作。



圖表一：自動議約的程序

CBB Model Stage	目標	人工議約	自動議約	本研究對應之解決方案
2.Production Brokering	找出可滿足需求的產品	從型錄或其他來源取得產品資訊和規格	透過線上目錄和 ontology 取得產品資訊和規格	Ontology System儲存產品資訊和規格
3.Merchant Brokering	找出可滿足需求的廠商	從工商名錄或其他來源取得廠商資訊	透過線上目錄取得廠商資訊	Facilitator儲存與特定產品相關之供應者和需求者資訊
4.Negotiate	進行議價簽訂契約	使用者到店家，評估產品價值，若滿意則完成交易，否則可根據經驗或策略來提出意見進行協商	由助理軟體透過網路，評估產品價值，若滿意則完成交易，否則可根據經驗或策略來提出意見進行協商	由助理軟體透過 Wrapper執行溝通工作，利用使用者的效用近似函數(NN)評估產品價值是否能使使用者滿意，若不行則可根據 Case Base所儲存的經驗或由Evaluator(GA)協助產生策略來提出意見進行協商

圖表二：人工與自動議約之比較

根據自動議約在供需之間進行的工作，本研究以元件化以及功能導向的觀點，提出一套自動議約系統的建構工具以及環境。本研究認爲一套良好的自動議約系統的建構環境需要有兩大功能，合作協商自動化以及助理軟體本身的評估決策的自動化。本研究以Agent EC-Wrapper以及Evaluator來達成此機制。Agent EC-Wrapper可讓實

作自動議約系統變的更快、更有效益，並且使助理軟體具有可重複使用性。而且透過Agent EC-Wrapper，也提供了助理軟體合作協商的自動化能力，使得實作助理軟體時，只需要專注於助理軟體本身的工作而不必花費太多的心思在溝通協定上。而使用Evaluator可從使用者觀點對所擁有的資訊進行評估，提供助理軟體自主能力。圖表二說明了議約的過程，以及本研究在議約的過程中所扮演的角色。我們將在下一節首先說明Agent EC-Wrapper的架構並於次節深入說明Evaluator。

2. Agent EC-Wrapper

Agent EC-Wrapper 基本假設

Agent EC-Wrapper 的目前是爲了提供助理軟體合作協商的自動化能力。而本研究設定了一些基本假設，以期此建構環境能達到我們的目標。Agent EC-Wrapper 的基本假設有下列幾點：

(1) 群體助理軟體的基礎環境應能將助理軟體本身運作策略與溝通能力分離。

將助理軟體運作策略與溝通能力分離可使助理軟體設計時不必考慮到溝通的問題。將現有的溝通協定以一較高層次的溝通協定代替，如此 Agent EC-Wrapper 就能替助理軟體完成所有溝通工作。

(2) 群體助理軟體的基礎環境應同時提供助理軟體合作與協商的能力

目前助理軟體的互動，可分爲合作(cooperation)以及協商(negotiation)兩種方式。合作表示助理軟體間有著共同的目標，並爲完成其共同目標進行合作。而協商模式則表示各個助理軟體間有各自利益的考量，並各自爭取自己的最大利益，所以可能發展出各種不同的協商策略。本研究認爲助理軟體是否需要與其它助理軟體合作，需視助理軟體的功能，發展環境應同時提供兩種機制。由於我們認爲，合作是協商的一種特例，因此將以協商機制爲主，但仍提供助理軟體的合作機制。

(3) 群體助理軟體的基礎環境應提供多重承諾(Multi level Commitment)的能力和考量溝通成本的問題

Sandholm & Lesser(1995)認爲助理軟體在允諾與其他助理軟體合作後，應能視本身情形決定是否使用悔約機制，也就是說他們認爲發展助理軟體時，應考慮允諾的層次、違約金等等相關議題。雖然考量允諾的層次將會使助理軟體之間的互動變的更複雜，但是 Andersson & Sandholm(1998認爲助理軟體考慮允諾的層次將可以增加整體效益。因此我們認爲是否考慮允諾的層次，應需視助理軟體的需要，發展環境不需做任何限制，它只需做好監督和認證工作。此外助理軟體與其他助理軟體互動時，可能會產生大量的訊息。同時若不設限助理軟體間的協商也可能永無止境。因此在互動的過程中應考慮溝通成本以避免類似問題發生。

(4) 群體助理軟體的基礎環境應考量到資訊或策略安全的保護，以及合約履行的問題

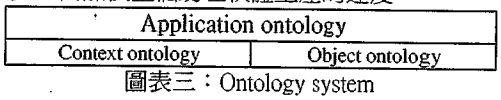
我們認爲一個自動議約系統也需要考慮下列安全性問題，以保護資訊或策略的安全，以及確保合約履行。

1. 助理軟體的識別和溝通的保密：群體助理軟體的基礎環境需要考慮溝通訊息是否會被僞造，以及是否會被竊取等問題，並且要能確認交易的真實性。
2. 合約的保證性：群體助理軟體的基礎環境需要能確實執行合約(含違約的懲罰)，確保合約的不可否定性，以及相關的付款問題。
3. 策略的保密性：助理軟體依目標的不同，可分爲許多類別，有合作型的，也有自私型的。群體助理軟體的基礎環境應提供過濾資訊的機制，以達到不同的策略的保密程度。

前二項爲電子商務的基礎建設，而目前已有許多成果，如 SET、SSL 和認證中心等機制存在，因此我們假設環境已提供一個可讓群體助理軟體運作的安全環境，本研究將著重於策略保密的機制。

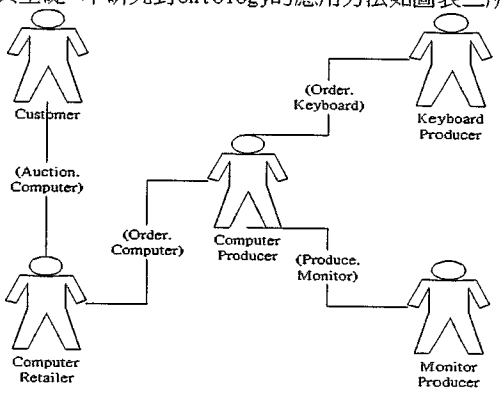
Agent EC-Wrapper 基本原理

以往開發助理軟體的設計者必須瞭解助理軟體進行合作的程序，而且以有限狀態機表現出助理軟體的具體合作溝通行爲。這表示助理軟體設計者必須將這些合作流程包含在助理軟體中。而本研究則希望能透過簡單的機制來達到合作協商程序的自動化，也就是說設計者並不需要將助理軟體與其他助理軟體合作協商的流程寫在助理軟體中，而是以簡單的方式來達到助理軟體的合作協商。助理軟體可透過Agent EC-Wrapper完成各種溝通工作，並可考量溝通成本、多重承諾、合作程度的處理以及擔任提供者或要求者的仲介等等。有了Agent EC-Wrapper可使得合作協商工作抽象化、一般化，原因在於助理軟體本身並不牽涉到合作協商工作的運作流程，而只透過Agent EC-Wrapper與其他助理軟體進行合作協商工作。如此就能將助理軟體與合作協商的問題減到最少，而加快整個助理軟體生產的速度。



圖表三：Ontology system

透過Agent EC-Wrapper，助理軟體不需要知道其他助理軟體位址，就能透過Agent EC-Wrapper的仲介，達到自動化動態服務尋找。也就是說我們希望助理軟體的關係能夠很動態的建立而且能即時反應這些動態的關係。當一個新的助理軟體發展出來之後，能夠很簡單的就被加入一個既有的群體助理軟體內。我們利用ontology system作爲連結助理軟體的媒介。Ontology即助理軟體與其他助理軟體互動時，需要有的一個對應用的共同體認與基礎。本研究對ontology的應用方法如圖表三所示。



圖表四：供應鍊系統的範例

助理軟體的應用是由一個application ontology所表示。而一個application ontology又是由context ontology和object ontology所組成。其中context ontology用來表達溝通協定的目的與行爲，如拍賣(Auction)、下訂單(Order)或生產(Produce)等等。而object ontology則是用來描述context ontology所要處

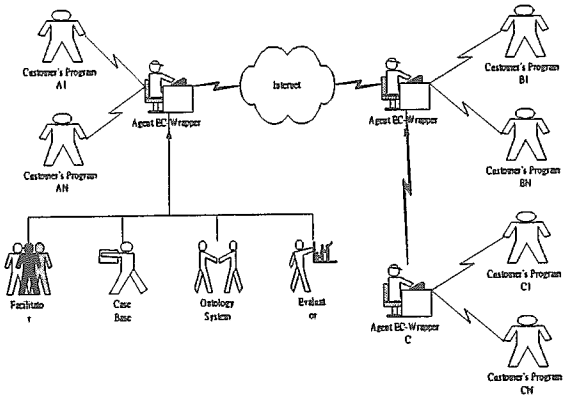
理的目的物，如電腦(Computer)等。Context ontology和object ontology的結合表達了某一特定層面的應用，如生產電腦(Produce. Computer)即爲一種application ontology。

在圖表四描述了一個簡單的供應鍊系統的ontology system。此系統共包含五個助理軟體，即Customer、Computer Retailer、Computer Producer、Keyboard Producer和Monitor Producer。其中Computer Retailer以競價拍賣(Auction. Computer)的方式販賣電腦給Customer。Computer Retailer可下訂單(Order. Computer)給Computer Producer。而Computer Producer可對其子公司Monitor Producer下達生產監視器的命令(Produce. Monitor)，或向其策略聯盟廠商Keyboard Producer以協商方式購買鍵盤(Order. Keyboard)。

如果Computer Producer不想再透過Computer Retailer販賣電腦，而想要進行直銷時，只要加上(Auction. Computer)的application ontology以及將(Order. Computer)去除，就會和Customer產生一個關聯並且中斷與Computer Retailer的合作，Customer和Computer Retailer完全不必做任何更動。同理，若有一個新的Keyboard Producer出現，並且也是使用(Order. Keyboard)的application ontology，則Computer Producer不需做任何修改也可以從新的Keyboard Producer取得服務。

在本例中，同時使用了拍賣(Auction. Computer)、合作(Produce. Monitor)和協商(Order. Computer)三種機制，而Computer Producer亦可將工程分包給另外兩個助理軟體。這顯示以此種方式來建構群體助理軟體，可以使設計者快速而有彈性的完成群體助理軟體的設計。

我們認爲考慮允諾的層次，違約金等等相關議題是群體助理軟體的基礎環境需提供的功能。在助理軟體與其他助理軟體互動過程中若考慮溝通成本，可以避免產生大量的訊息，或長久協商的問題，而考慮違約金可增加整體利益。本研究將違約金和成本及其他的資訊都視爲一種屬性來實作，並且儲存於Ontology Base中。透過相關屬性的組合應用，將可提供助理軟體在策略設計的多樣與彈性，例如在協商過程中，對溝通次數的限制等等。



圖表五：Agent EC-Wrapper 的架構

Agent EC-Wrapper 的架構

本研究提出的Agent EC-Wrapper乃由數個子元件所組成，各元件功能分述於下。並在本小節最後說明了Agent EC-Wrapper的使用範例。Agent EC-Wrapper子元件說明如下：

(1) Facilitator

在本架構下，ontology的建立者(即儲存ontology的那個Agent EC-Wrapper)，將是所有使用同一個application ontology的助理軟體的中心。助理軟體透過中心Facilitator的運作，就找到其相關聯的助理軟體。並且，各個獨立的助理軟體也會由於關聯的產生而形成一個群體助理軟體。也就是說我們運用Facilitator和ontology system的結合，提供仲介的服務。一個群體助理軟體必需至少有一個中心，但是並不限定於一個，因爲群體助理軟體也是能由其它的群體助理軟體所組成，如同網際網路的觀念。

在ontology建立者的Facilitator儲存了使用特定application ontology的應用提供者(Provider)和需求者(Requester)的資訊。在使用Agent EC-Wrapper於助理軟體時，首先要向需要使用到的ontology的建立者取得ontology資訊(取得ontology的工作仍需由人介入)。在助理軟體設計者根據ontology的規格設計助理軟體後，需對ontology的建立者進行註冊工作(助理軟體可由ontology的資訊找出ontology建立者)，在將助理軟體所用的application ontology和助理軟體所扮演的角色傳送給ontology建立者的Facilitator後，助理軟體就連結上一個群體助理軟體。

以供應鍊的例子說明，Computer Producer爲Produce和Order的ontology設計者，Computer Retailer爲Auction的ontology設計者。Computer Retailer若希望以(Order. Computer)的application ontology和外界連結，就必需向Order ontology的設計者Computer Producer的Facilitator註冊，以建立連線。另外，Monitor Producer使用了Produce ontology，而Produce也是由Computer Producer所定義，因此Monitor Producer會透過Computer Producer的Facilitator與外界連結，提供服務給相關的助理軟體。

Facilitator		
Agent Name	Computer Retailer	
Application Ontology	Agent	Role
Auction. Computer	Customer	Requestor
Auction. Computer	Computer Retailer	Provider

Facilitator		
Agent Name	Computer Producer	
Application Ontology	Agent	Role
Order. Computer	Computer Retailer	Requestor
Order. Computer	Computer Producer	Provider
Produce. Monitor	Computer Producer	Requestor
Produce. Monitor	Monitor Producer	Provider
Order. Keyboard	Computer Producer	Requestor
Order. Keyboard	Keyboard Producer	Provider

圖表六：Facilitator 範例

Context Ontology			
Name	...	Default Action	...
Auction	...	Satisfied_Utility=Initial_Utility; if (Negotiating) (if Proposal_Utility>Reserved_Utility Deal() else if Satisfied_Utility< Reserved_Utility Fail(); Decrease(Satisfied_Utility); RFB(); ...)	...

圖表七：Context ontology 範例

由於本例所用到的ontology分別儲存於兩個助理軟體，因而會使用到兩個Agent EC-Wrapper的Facilitator(每個EC-Wrapper都有Facilitator，但不一定都要用到)。使用者亦可根據需求對ontology作集中式的管理，建立一專責的助理軟體，負責ontology和Facilitator的管理，以維持ontology的標準化，如同eCo System或Ontolingua所扮演的類似角色。

(2) Ontology Base

Ontology Base是儲存ontology system的各種規格，作爲溝通的基準。其中，context ontology用來表達溝通協定的目的與行爲，object ontology則是用來描述context ontology欲處理的目的物，application ontology則描述了應用的運作策略和處理模式等資料。以(Auction. Computer)爲例，圖表七爲拍賣協定(Auction)的context ontology。除了各種資料和規格外，亦可在此描述協定的預設運作方式。

而圖表八爲電腦(Computer)的object ontology。除了基本資料外，亦描述了Computer具有的屬性，以及屬性的特性。第3.2節提到的運作機制，如成本或違約金，都可在此定義。而爲了加速效用近似函數(詳細資料請見第四節)的建構速度，亦可在此定義一些內定的效用近似函數。使用者可從預設的角色的資料做修正，建構個人的效用近似函數。

Object Ontology						
Name		Description				...
Computer		This is Computer's ontology				...
Attribute	Type	Interval	Value	Default utility approximator		
Price	Number	1000	40000 50000	Type	Description	Weight
RAM	Number	32	32 256	Students	Price -oriented	1,0,0,...
CPU	Symbol	0	486 586 686	Game Player	Performance -oriented	0,1,0,...
Penalty	Expression	0	Price 0.05	3D Designer	All covering	0,1,1,...
Times	System	0	0	Normal	Average	0.5,0.5, 0.5, ...
...	...	...	...	...	...	...

圖表八：Object ontology 範例

和前述的context ontology和object ontology不同的是，application ontology儲存在每個Agent EC-Wrapper的Application Ontology Base中。Application Ontology Base是儲存了助理軟體對其使用到的application ontology所需要的資訊、處理方式和運作策略。如圖表九所示，Computer Producer使用了三個application ontology，分別以三個程式處理。當收到屬於某個application ontology的資訊時，就以對應的Customer Program處理。而此處的處理程序也是使用者需要自行設計的部份(使用者亦可使用Ontology Base中所定義的內定處理程序，請見圖表七)。有關Customer Program的設計，此架構不預設使用者需使用何項語言工具，而是以Agent EC-Wrapper做爲Customer Program的通訊代理軟體，即Customer Program透過呼叫Agent EC-Wrapper的功能以達成其目的。而在Policy項目則規範了助理軟體的行爲模式，包括資訊的過濾，策略的透明度等等。以過濾輸出的資訊的方式，產生不同的溝通模式，例如使用者若選擇開放使用者資訊的輸出，則商家可以進行各種客製化的服務。然而使用者若選擇關閉所有使用者資訊的輸出，使用者亦可進行匿名交易。

Application Ontology Base			
Agent Name	Computer Producer		
Application Ontology	Customer Program	Policy	Utility Approximator
Order. Computer	Program_OC	Selfish	Approximator_OC
Produce. Monitor	Program_PM	Friend	Approximator_OM
Order. Keyboard	Program_OK	Cooperation	Approximator_OK

Application Ontology Base			
Agent Name	Customer		
Application Ontology	Customer Program	Policy	Utility Approximator
Auction. Computer	Program_AC	Selfish	Approximator_AC

圖表九：Application Ontology Base 範例

(3) Evaluator and Case Base

Evaluator是用來達到助理軟體的自動化，它仍是Agent EC-Wrapper的一部份，但由於較其他子元件重要且複雜，因此另做一節討論。而系統在達成協議後，會將協商過程記錄在Case Base內，以便日後同一ontology的協商能很快的達成協議。Case Base將和Evaluator一併討論。

範例

助理軟體的功能，爲有智慧且自主的爲使用者完成特定的工作。在本架構下，Agent EC-Wrapper的角色就是助理軟體的助理軟體，Agent EC-Wrapper將會有智慧且自主的爲助理軟體完成相關的溝通工作。例如，要設計一個Customer助理軟體使其具有購買電腦的能力，可以下列一行指令表示開啓協商。

```
BuyComputer=Wrapper.CreateApplicationSession("Auc
tion","Computer",Specification);
```

而Agent EC-Wrapper將透過Facilitator找出那些助理軟體有提供販賣電腦的服務，並自動取得連繫，並要求提出提案。由圖表六中可看出角色爲Provider，而提供(Auction. Computer)服務的助理軟體有Computer Retailer。Customer即透過Agent EC-Wrapper對所有提供販賣電腦服務的助理軟體發出要求提案的訊息。當Customer收到Computer Retailer送出的提案時，Customer的Agent EC-Wrapper就從Application Ontology Base中找出對應的處理程式。由圖表九中我們得知，Agent EC-Wrapper將會執行Program_AC來處理由Computer Retailer送來的提案。而對於像English Auction這種常用的協定，Customer亦可直接使用定義在context ontology中的內定處理程序。在圖表七中的Default Action項目中，即爲English Auction的處理程序，使用者只需給定Initial_Utility和Reserved_Utility即可使用。

在處理程序中若要對提案進行評估，就是使用Application Ontology Base中的效用近似函數項目，approximator是預先建構儲存使用者喜好的類神經網路，可將提案進行量化，以便進行協商。若在協商過程中，使用者需要取得特定資訊，都可以透過object ontology中描述的規格資訊向Agent EC-Wrapper取得。假設我們需望協商過程不超過10次，以節省網路成本，則可以下列指令控制協商次數在10次以內。

if(Wrapper.GetInformation("","Computer","Times",Buy

Computer)>10)Wrapper.Fail(BuyComputer);

因此只要將各種特定資訊加入ontology system中，將可形成各種特殊的應用。例如將多重承諾的觀念加入，就可允許助理軟體在付出一罰金後毀約，以達到整體的利益。在圖表八中則定義了Penalty項目，明訂使用者在訂購電腦後若要反悔，則要付出售價的百分之五作爲違約金。

3. Evaluator

Evaluator是用來使助理軟體自動化，其主要功能爲評估外界的提案(proposal)，並且根據使用者喜好做出回應。其中最難處理的就是多重屬性價值的評估(multiattribute evaluation)。以往的多重屬性協商的作法，大多是以效用函數來處理此類問題，如此即可將多重屬性的問題視爲單一屬性，就如同以價格作爲協商因素。但此種作法有一些問題，例如各個屬性的權值沒有一個好的科學的方法可以決定以及無法處理各個屬性之間的交互影響，以上文所提到的Computer爲例，以其效用函數爲下式。

$$U(Computer)=W_p*U_p(Price)+W_r*U_r(RAM)+W_c*U_c(CPU)$$

對使用者而言，他無法很容易的找出他自己對這三個屬性的效用函數，亦無法決定每個效用函數的權值，更無法決定各個屬性間對使用者的交互影響。例如對一個特別重視電腦記憶體大小的使用者而言，RAM很大時他對Price的效用函數可能與RAM不足時對Price的效用函數有很大的差異。

由於上述的問題使得我們認爲要使用者客觀的決定效用函數是困難的，但如果單純要使用者依其喜好決定某屬性組合的好壞則是比較可行的。因爲使用者可以直覺的決定某種屬性組合他是否願意接受；或者兩種屬性組合中，他較喜歡那一種，甚至加以排名，即可找出使用者的效用函數，本研究將以類神經網路(neural network)的方法解決這問題。將屬性組合當成輸入節點，而使用者對該屬性組合的接受程度視爲輸出，建構一個類神經網路，來當作一個效用函數的approximator。我們選擇使用類神經網路的原因是：

1. 類神經網路中，各個節點的權值是由樣本資料中找出來的，避免了需由使用者指定屬性權值的問題。使用者只要表達對樣本資料的喜好情形。
2. 根據Cole & Muthusamy(1990)和Shavlik, Mooney & Towell(1991)的研究，類神經網路可處理各輸入節點間任何線性、非線性的關係。如此就能處理各個屬性之間的交互影響。

因此只要使用類神經網路，就可解決權值設計和非線性影響的兩個問題。

Evaluator基本假設

Evaluator希望以類神經網路的特性，找到使用者效用函數的approximator，去除需要決定屬性權值及交互影響的困難。同時，Evaluator再以基因演算法(Genetic Algorithm)，將類神經網路所形成的效用近似函數作爲fitness function，來找出相對的回應。

因此本研究的Evaluator運用時有一假設：若使用者(或系統)能提供類神經網路足夠且具代表性的樣本(training sample)，則產出的類神經網路就能充分反應使用者的喜好。本研究由兩個方向取得使用者喜好的樣本資料：一部分是由使用者提供其喜好資料；另一部分是由過去成交案例取得，這就是Case Base所希望提供的功能。

Evaluator基本原理

Evaluator主要的基礎就是將類神經網路類當成效用近似函數(utility approximator)，如此就可以單一屬性的協商方式處理。以下介紹Evaluator的運作方式。

(1) Training

此一階段由系統根據object ontology產生一些多重屬性組合的樣本，並由使用者給予評價。在此仍以前面的例子說明，如圖表十所示，假設application ontology是(Auction, Computer)，而Computer有Price、RAM、CPU等三個屬性。則系統將爲Customer產生圖表十所顯示的樣本：其中的效用是由使用者依其喜好所給定的，其意義可以是排名或分數，由使用者賦與其意義。此樣本經由類神經網路即可產生該名使用者對Computer這物品的效用近似函數。使用者亦可直接使用系統提供的內定值(請見圖表八)，加快效用近似函數建構速度。

(2) Negotiation

在協商之中當對方提出一個提案，如(45000, 64, 586)，Evaluator就可利用效用近似函數算出提案的效用。當對方提案的效用達到可接受的程度(此例要求效用需大於零)，此筆交易就算成交。而如果對方提案的效用無法達到可接受的程度，如本例的效用爲-7，則Evaluator將提出修正提案(Counter Proposal)進行協商。

Customer			
Price	RAM	CPU	Utility
40000	128	686	7
45000	128	686	2
40000	64	686	-1
40000	64	586	-2
45000	64	686	-6

圖表十：效用近似函數的訓練資料

(3) Generating Counter Proposal

如同價格的協商方式，在提出修正提案時，也將從效用最高的屬性組合開始提出，爾後逐步下降，直到成交爲止。根據Matos & Sierra(1998)的研究指出，調整效用的方式有Time-dependent、Resource-dependent和Behavior-dependent等等許多方法可供應用，使用者可依據需求，選擇適用的機制。此外，要求調整效用需遵循一原則(proper quantum principle)[27]，以避免進行過度的協商。此限制要求己方提案變化量需達到對方提案變化量的一定比例。以圖表十一爲例，對Customer而言，Computer Retailer提出提案的效能由-7升爲-6，變化量爲1。根據原則，當Customer要提出修正提案時，變化量至少爲1，因此Customer必需提出一個效能爲6的提案。

而提出修正提案的方式，是以基因演算法在限定的效用下找出最佳的屬性組合。例如在本例中，若Customer提出(40000, 128, 686)這個效用爲7的修正提案對方無法接受，Evaluator將利用基因演算法找出效用爲6的修正提案，如(40000, 128, 586)即爲一解(本例原是English Auction機制的應用，Customer不會提出修正提案，在此我們爲方便說明協商過程，因爲允許Customer可以提出修正提案)。

(4) Using Case Base

圖表十一爲此例協商並達成協議的過程。首先

Computer Retailer提出(45000,64,586)，這個提案對Computer Retailer而言，具有的效用爲3。但對Customer而言，卻是-7，因而無法接受，Customer即可提出修正提案。此例在五個步驟下即達成協議，但事實上協商過程可能會很冗長並且提案很難有交集，如[9]和[20]的協商次數從20到10^{10}次不等。因此系統會將達成協議的過程記錄在Case Base內，以便日後同一ontology的協商能很快的達成協議。本研究預期能透過Case Base，以下列兩種方法加快協商的速度：

1. 當Case Base累積大量的案例後，Computer Retailer可以從Case Base中找出Customer的效用近似函數，經過事先計算後，直接提出最接近協議的提案，以加快協商過程。
2. 使用者越重視的屬性，應會是在協商過程比較後面的部分才妥協的屬性，或者是妥協程度較少的屬性。如圖表十一所示，Computer Retailer如果透過Case Base，應可發現Customer對RAM有某種程度的堅持，若Computer Retailer能因此針對Customer對RAM的執著而提出修正提案，協議也可較快達成，並可滿足使用者的偏好。

	Customer		Computer Retailer	
#	Proposal	Utility	Utility	Proposal
1		-7	3	(45000,64,586)
2	(40000,128,686)	7	-5	
3		-6	2	(45000,64,686)
4	(40000,128,586)	6	-4	
5	Deal	1	1	(45000,128,586)

圖表十一：協商的範例

4. 應用

本研究提出的是一個建構自動議約系統的建構環境，使用者可依需求建構群體助理軟體。我們認爲凡是具有供需角色的應用，都可透過 Agent EC-Wrapper，進行自動議約，以滿足供需雙方的需求，本節將提出幾個本工具可能的應用。

Agent-based Supply Chain

利用助理軟體技術來形成供應鍊的研究已有許多成果了，例如 ISCM[11]、CIIMPLEX[19]等。而依據 Fox, Chionglo & Barbuceanu(1993)的研究，一個助理軟體-based supply chain system 應該具有某些特性。有些是原本群體助理軟體就具備的特性，如 Distributed, Dynamic, Interactive, Anytime, Complete, General, Cooperative, Intelligent, Responsive, Reactive 等。而某些特性，如 Integrated, Reconfigurable, Adaptable, Backwards Compatible，卻可由本研究所提出的方法協助達成。如文中的供應鍊範例，透過 ontology system 的設計和連結，可以很快速的形成一供應鍊，而日後若要變更流程或是增減供應鍊的成員，都可在不更動原有系統的狀況下快速完成。

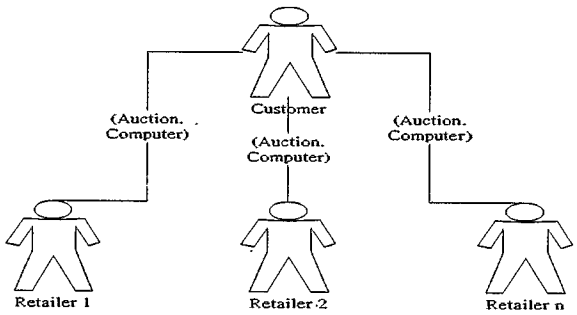
CBB Model-based Agent

Guttman, Moukas & Maes(1998)所提出的消費者購買行爲模型是解釋消費者購買產品或服務的行爲模式。圖表十二爲消費者購買行爲模型的內容以及對目前幾個著名的電子商務應用網站的分析。

Stages \ Examples	Persona Logic	Firefly	Bargain Finder	Jango	Kasbah	AuctionBot	Tete@Tete
1.Need Identification							
2.Production Brokering	X	X		X			X
3.Merchant Brokering			X	X	X		X
4.Negotiation					X	X	X
5.Purchase and Delivery							
6.Product Service and Evaluation							

圖表十二：Consumer buying behavior model with agent mediation [16]

本研究提出的工具可以快速建構符合消費者購買行爲模型的應用系統。以購買電腦爲例，利用(Buy.Computer)的 application ontology 連結供需雙方的助理軟體。在消費者購買行爲模型第一階段爲需求確認，Evaluator 可以依使用者喜好，定義出使用者的效用近似函數，爾後行爲皆爲根據使用者需求進行。而之後商品仲介、商家仲介和協商三個階段，在本架構下並無明顯分界，使用者選用了某個 application ontology 後，即與其他使用同一 application ontology 的助理軟體產生連結，而在協商時，助理軟體依使用者的效用近似函數送出產品規格後，相關的零售商即可依規格做出相對的回應，這三個階段持續進行直到產生結果，完成使用者的目的。



圖表十三：CBB 模式的群體助理軟體範例

Enabling Business Transformation

Dutta & Segev(1999)從行銷的角度去探討企業在網路市場(Marketspace)中成功轉型的因素。由資料得知目前企業轉型的程度仍然偏低，而一個自動議約系統可以有效的直接或間接的完成其中大部份的工作，加速企業轉型。例如 online pricing information、dynamic customization of prices、online price negotiation、online ordering、real time processing of orders、the involvement of partner organization in online distribution、online production information等等工作皆已在前文的例子中說明過。而其他許多的工作也可透過 ontology system 的設計或是助理軟體的策略運用而完成，如 the provision online communications to customers、the solicitation of online feedback from customers、the customization of products、the participation of customers in the specification and design of products等工作。我們認爲，借由支援上述工作的完成，將可協助企業的轉型。

5. 結論

關於自動議約類型的工作，越來越多，對於支援自動

議約機制的需求也日異重要，但這方面的工作卻因爲牽涉到 ontology 和溝通協定的標準化，以及自動化的協商機制，因此實用上有諸多困難。因此我們提出一個自動議約的解決方案。以一個 Agent EC-Wrapper 將助理軟體包裝，使其具有與外界同類型(即具有相同的 ontology 基礎)的助理軟體溝通的能力，使助理軟體設計者可以將重心著眼於助理軟體的功能與策略設計，使助理軟體的設計具有元件化、再用性等特性，以達到快速建構群體助理軟體的目的。而對於協商工作中的評估工作，我們以類神經網路作爲效用近似函數，由於類神經網路的特性，使得我們的評估方法具有多屬性、直覺化、及非線性問題的處理能力，得以協助協商工作的進行。

對於本研究的適用範圍，我們認爲不管是自動議約系統或是供應鍊系統，這些系統都有一些特性。一個問題若能將其運作模式歸納爲下列這三種特性，本架構皆能有效支援。

1. 具有供需兩個角色，進行溝通以滿足需求
2. 收集即時資料並且評估所擁有的資料
3. 依據情評估情形做出行動

本研究是爲支援前兩項工作而 設計(第三項工作爲助理軟體的功能與策略)，Agent EC-Wrapper 用來快速的建立助理軟體間的溝通機制，而以 Evaluator 作爲評估環境的支援工具。因此各種自動議約的應用此架構皆能有效的支援。

我們同時亦希望，自動議約系統的建立，不只是加強群體助理軟體的功能或是增加應用的方式，更能從改變企業程序的方向著手，加速企業轉型，甚至於協助建構虛擬組織或企業，以因應這個快速改變的虛擬市場。

參考文獻

1. Andersson, Martin R. and Sandholm, Tuomas W., "Leveled Commitment Contracting among Myopic Individually Rational Agents", Proceedings of the Third International Conference on Multiagent System, 1998.
2. AuctionBot, "Michigan Internet AuctionBot", http://auction.eecs.umich.edu.
3. Beam, Carrie and Segev, Arie, "Automated Negotiations: A Survey of the State of the Art", CMIT Working Paper 97-WP-1022, May 1997.
4. Bichler, Martin, Kaukal, Marion and Segev, Arie, "Multiattribute auctions for electronic procurement", Proceedings of the First IBM IAC Workshop on Internet Based Negotiation Technologies, March, 1999.
5. Baum, M. S. and Perrit, H. H., "Electronic Contracting, Publishing, and EDI Law", John Wiley & Sons, 1991.
6. Chavez, Anthony and Maes, Pattie, "Kasbah: An Agent Marketplace for Buying and Selling Goods", Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, April 1996.
7. Cole, L.A. and Muthusamy, Y., "A Performance Comparison of Trained Multilayer Perceptions and Trained Classification Trees", Proceedings of IEEE, Vol.78, 1990.
8. Dutta, Soumitra and Segev, Arie, "Transforming Business in the Marketspace", Proceedings of the 32nd Hawaii International Conference on System Science, 1999.
9. Dworman, Garett, Kimbrough, Steven and Laing, James, "On Automated Discovery of Models Using Genetic Programming in Game-Theoretic Context", Journal of Management Information Systems, Winter, 1996.
10. eCo, "The eCo Framework, a CommerceNet Project", http://eco.commerce.net/,
11. Fox, Mark S., Chionglo, John F. and Barbuceanu, Mihai, "The Integrated Supply Chain Management System", 1993
12. Glushko, R. J., Tenenbaum, J. M. and Meltzer, B., "An XML Framework for Agent-based E-Commerce", ACM, 1999.
13. Gruber, T., "Ontolingua: A Mechanism to Support Portable Ontologies", Stanford University, Knowledge Systems Laboratory, Technical Report KSL-91-66, March 1992.
14. Guttman, Robert H., Moukas, Alexandros G and Maes, Pattie, "Agent-mediated Electronic Commerce: A Survey", Knowledge Engineering Review, June 1998.
15. Guttman, Robert H. and Maes, Pattie, "Agent-mediated Integrative Negotiation for Retail Electronic Commerce", Proceedings of the Workshop on Agent Mediated Electronic Trading, May 1998.
16. Guttman, Robert H., "Agent-mediated Integrative Negotiation for Retail Electronic Commerce", http://guttman.www.media.mit.edu/people/guttman/research/commerce/talk10/index.htm, Presentation of the Workshop on Agent Mediated Electronic Trading, May 1998.
17. Maes, Pattie, "Modeling Adaptive Autonomous Agents", Artificial Life Journal, edited by Christopher G. Langton, MIT Press, 1995.
18. Matos, Noyda and Sierra, Carles, "Evolutionary Computing and Negotiating Agents", Proceedings of the Workshop on Agent Mediated Electronic Trading, May 1998.
19. Peng, Y., Finin, T., Labrou, Y. and Cost, R.S., "An Agent-Based Approach for Manufacturing Integration - The CIIMPLEX Experience", International Journal of Applied Artificial Intelligence, Vol.13 No.1-2, 1999.
20. Oliver, Jim R., "A Machine Learning Approach to Automated Negotiation and Prospects for Electronic Commerce", Journal of Management Information Systems, Vol.13 No.3, 1996.
21. Ontolingua, "Stanford KSL Network Services", http://ontolingua.stanford.edu.
22. Runge, Alexander, "The Need for Supporting Electronic Commerce Transactions with Electronic Contracting Systems", Proceedings of Conference on Information Systems and Technology, April 1998.
23. Sandholm, Tuomas and Lesser, Victor, "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework", First International Conference on Multiagent Systems, 1995.
24. Shavlik, J.W., Mooney, R.J. and Towell, G.C, "Symbolic and Neural Net Learning Algorithms: An Empirical Comparison", Machine Learning, Vol.6, 1991.
25. Uschold, Mike, King, Martin, Moralee, Stuart and Zorgios, Yannis, "The Enterprise Ontology", The Knowledge Engineering Review , Vol.13, Special Issue on Putting Ontologies to Use, 1998.
26. WordNet, "WordNet - a Lexical Database for English", http://www.cogsci.princeton.edu/~wn/online/.
27. Wu, Shih-Hung and Soo, Von-Wun, "Game Theoretic Reasoning in MultiAgent Coordination by Negotiation with a Trusted Third Party", Proceedings of Autonomous Agents, 1999.
28. Wu, Zeng-long, and Yuan, Soe-Tsyr, "Automatic Agent Cooperation Wrapper", Proceedings of Agent Technology Workshop, 1999.