

本體論技術於智慧型家庭監控系統之應用

Ontology-Based Modeling of Security Control for Smart Home Application

劉智漢

國立政治大學資訊科學系

Email: andyliu@gate.sinica.edu.tw

李蔡彥

國立政治大學資訊科學系

Email: li@nccu.edu.tw

摘要—隨著 ontology 應用及語意網的普及，可以預見將有越來越多的應用系統以 ontology 做為異質系統間之資料交換格式。若現行的分散式監控系統，也能夠以 ontology 做為程式外部的溝通及資料描述介面，那不同的監控系統彼此要交換資料就會更容易。此外，系統在面對裝置異動或邏輯修改時，也能更快速的面對異動。本研究的目標，便是想要從監控系統中分離出程式應用邏輯及設備描述邏輯，改以 ontology 的描述方式來取代，進而建立一套以 ontology 為核心的監控系統，並藉由 ontology 的導入來改善現行監控系統修改不易及彈性不大的缺點。為了驗證上面的論述，我們設計了一套以 ontology 為組態描述的 webpanel 系統。製造商可以用符合 ontology 的語法來描述設備的規格，系統整合商也能透過 ontology 設計監控系統的應用邏輯，結合兩者的 ontology 描述，便能讓 webpanel 系統執行監控程序。在本論文中，我們以實例說明 ontology 做為監控系統之應用邏輯描述的可行性。

關鍵詞—Ontology Design, Security Control, Smart Home

一、簡介

由於網際網路的發展，不同區域的監控系統有了一個整合的契機，讓使用者透過網路便可以觀看遠距的影像甚至透過網路來遠端控制設備；也因為這些原因，使得監控設備紛紛加上了網路連結的功能，讓原本獨立運作的監控設備有機會與其他設備溝通並產生更多的應用，也連帶使得分散式監控漸漸變成未來趨勢。但現有的監控系統大多是依據監控設備的特性及使用者的需求去開發軟體，缺乏動態整合的彈性。

程式設計師多依據不同設備的特性，開發各種適合當時流程及設備特性的監控系統。例如，機房環境監控系統、電力監控系統等。每套不同的監控系統都各自有各自不同的程式撰寫方式及邏輯，這些適合特定環境及設備的程式很難在

日後的新系統中被重複使用。系統上線後，也可能因為人員異動或文件不足，以至程式無法繼續再維護。此外，系統運作期間有些設備可能會損壞，在設備維修的期間，是否能夠快速的找到替代的設備而不去更動監控系統的程式，也是一個挑戰。每套監控系統完成後，在建構過程中所累積下來的知識，也常隨著時間消逝、人員流失而無法累積傳承，非常可惜。

分散式監控漸成趨勢，要如何整合應用管理分散各地的監控設備也是一個值得思考的問題。因此，本文的著眼重點便在於如何抽離出寫死在程式中的設備功能及組態、應用邏輯及反應流程，改用 ontology 方式描述，讓程式邏輯能夠從軟體黑箱中被抽離出來，以便使用者能夠用簡易的描述及理解的概念來記述這些邏輯，進而讓監控系統的核心程式能被重複使用，程式設計師只需專注在核心程式的維護；至於反應流程及邏輯的部份，就交由系統整合商去依據實際需求定義。我們希望如此便能有效的減少系統開發及維護修改所需要的時間及成本。

本研究提出將設備功能及應用程式邏輯從系統軟體中抽離出來，透過 ontology 來描述設備的功能、反應流程及監控邏輯；同時讓系統整合商能夠很快速方便的透過 ontology 描述檔來表達他們想要達成的監控流程。為了達到此一個目標，我們實作出一個稱為 webpanel 伺服器，用以處理及執行使用者定義出來的 ontology 描述檔，以便這些 ontology 能夠被 webpanel 解析並轉化為真正的監控流程；進而根據這些流程來啟動／監控監控區域的各個感應器及感應器回傳的訊息，並依據監控 ontology 事先定義的反應方式，對不同的訊息做出不同的回應及通知。

雖然 ontology 目前已經被許多領域的專家學者使用，但在監控的部分，卻較少有研究導入 ontology 的觀念來將監控流程抽離出監控軟體，讓監控流程更加具有彈性及更容易客製化。

二、相關研究

目前已有許多 Web 服務技術(Web Service) 與 ontology 結合所建構出來的自動監控或智慧家庭應用。例如，在[2]這個研究中，作者提出的系統可以讓使用者透過語音介面將命令傳送給 Spoken Dialogue Agent(SDA)，而 SDA 則透過 SOAP 協定到家庭資訊系統服務的註冊點(UDDI)去做查詢，就可以知道哪些設備提供哪些服務，進而將這些服務提供給使用者。在 ontology 資料的建立方面，他們是採用了自訂的 MIML 格式的 XML 檔案，來對設備、功能及狀況做輸入及輸出的描述語言，與其他系統交換資訊則需要使用符合其系統所定的 XML。

另一方面的研究則是以 Agent 為基礎所建構的分散式服務。例如在[5]中，作者透過分散在各地的觀測設備將其觀測蒐集到的資料回傳到集中管理的中控中心（稱為 Information Broker (IB)），再由 IB 整合各種資訊後將這些資訊交由具有特定功能各司其職的 Agent 去做判斷是否要採取相關措施。這是另一種結合 Agent 與 ontology 來建構出跨異質設備的監控系統，但由於此系統採用的是由 FIPA 或其他 agent 系統所使用的 DAML ontology[8]尚不足以負荷複合的服務，因此要如何透過 ontology 的描述去解決各個異質設備服務間的整合，便是其未來的挑戰。

在[6]的研究中，作者以網站為基礎，結合 TCP、XML、HTTP 等開放技術為主，使用資料庫做為儲存介面，將使用者控制家電裝置的介面設計轉換為存取網頁資料庫的設計。此研究意圖讓具有網頁設計能力的使用者，也可以親身參與設計，透過網路輕鬆存取控制家中家電裝置的研究。

[1]則是另一種以 ontology 結合網路的應用。主要在探討以 ontology 為基礎的旅遊資訊擷取系統，希望能夠透過旅遊網站代理人技術，協助網站提供正確的資訊供使用者做出旅遊計

畫，進而提高旅遊產品的銷售機率。但此論文目前僅提供資訊系統架構，實驗部分則尚未舉實例來驗證。

由以上研究可以發現，ontology 的研究已經深入在各研究領域中，而網路結合 ontology 也成為重要的研究方向。然而，在我們尋找相關資料的過程中卻發現，ontology 在監控系統及智慧家庭的研究中卻是相對較少，但 ontology 的語言特性卻非常適合用來將真實世界的感知轉化成抽象概念的描述，以將監控系統中的程式邏輯通用化。因此，本研究將嘗試以 ontology 的方式探討如何將資料與應用描述邏輯抽離系統程式，以達系統彈性及重用的目標。

三、Ontology 本體論介紹

Ontology 一詞源自哲學上的「本體論」，是一套用來解釋存在意義的理論。由於近年語意網路的興起，讓 ontology 在電腦界的應用也日益增多，例如：語意部落格編輯器(semiBlog Editor)[9]，可以讓使用者輕易的在企業內部交換電腦中的資訊；也有使用 ontology 作為監控恐怖主義份子網路訊息的預警系統[4]，此外，如知識管理(knowledge management)、電子商務(electronic commerce)、自然語言研究(natural-language research) 等，也都嘗試將 ontology 導入以作為相關應用。由此可見，ontology 的應用在現今的電腦界中可說是非常廣泛。

而現實中也由於人們對相同的事情會因為不同的生活背景經驗而有不同的描述方式，也由於這個緣故，XML 結合 URI 的出現反而解決了這個問題。使用 XML 每個人都可以有自己的描述方式，但卻可以透過 URI 去知道不同描述所指的是相同的事物。但由於 XML TAG 並不能表達語意(Semantics)因此，必須要透過一些事先蒐集定義好的資訊(collections of information) 來賦予 XML TAG 意義，以便人與機器能夠透過它來了解彼此所指的意義，而這個事先蒐集定義好的資訊，我們便把它稱為 ontology。

Ontology 藉由提供某個研究領域的相關詞彙搭配 URI 的概念來解決不同人用不同詞彙描述相同事情的問題，適合用來做為人與機器溝通

的中介語言。舉例來說，當我們要描述門窗保全系統時，由不同的程式設計師來描述就會有不同的描述方式，程式的寫法也會因人而異，若改以 ontology 作為邏輯的描述語言，儘管不同的人使用不同的描述方式，但透過 ontology 表達之後，系統可以知道他們描述的事實上是同一個概念及關係，便能依據 ontology 描述去了解整個程式運行的邏輯進而讓機器執行相關的程序。

因此，ontology 可以用來為特定領域提供一套共享及可重複使用的知識分類架構，以做為人與機器或機器與機器相互溝通的媒介。透過 ontology 我們可以將複雜、需要推理、或是非系統化的資訊以明確的方式表達出來，非常適合用來做為不同網站之間彼此了解彼此 XML 標籤所代表的意義。也由於 ontology 具備上述的特性，因此我們此次的研究便採用 ontology 作為 webpanel 描述檔的核心，以求得人與機器，機器與機器互相溝通的效能以及資訊完整性。

前面有提到，由於 XML 的標籤在製定上較自由，常會導致一件相同的事或物，由不同的人來描述便會訂出不同的標籤，導致混亂而不利於機器閱讀。同時，XML 在抓取資料的設計上是採用 XPath 路徑表達式的方法來攫取資料，會存在對語意的解釋不完整的問題，也會造成資料讀取時不完整的問題發生。例如，「講師是大學機構成員的子集合」這個句子表示講師是大學機構的成員之一。這邊連接講師與大學機構的，便是子集合 (a subclass of) 的觀念。例如，假設有個 XML 是用來描述大學裡目前的狀態：

```
<大學機構成員>張小明</大學機構成員>
<教授>李小四<教授>
<課程 name=" 語意網與資訊網服務" >
  <開課老師>胡小三</開課老師>
</課程>
```

假設我們想要找出大學機構成員有那些人，透過 XML 的 XPath 的方式來抓取資料，我們只能得到：張小明這個答案，因為 XPath 的抓取資料方式是使用 <大學機構成員>tag，符合的條件只有一個。但從人類的理解，大學機構成員的答案會有三個，分別是：張小明，李小四教授與胡小三老師。為什麼會有這麼大的差距呢？因

為

- 所有的教授都是大學機構成員
- 大學課程是能由該所大學教授來開課

上面的規則加上人類的推理我們可以知道答案絕對不會只有 XPath 所找出來的一個答案。也由此我們可以發現到 XML 在描述事物時，除了 tag 之外，還必須要有其他的規則及輔助語言，才能將事物做完整且有意義的描述，而語意便可以透過 ontology 的描述來達成。就像 Tim Berners-Lees 在 “Semantic Web Road map” [3] 簡介中所提到的：

“The Web was designed as an information space, with the goal that it should be useful not only for human-human communication, but also that machines would be able to participate and help.” 除了人與人溝通之外，最重要的還是要機器也能夠參予並提供協助，這樣才是一個好的中介語言，也是 ontology 希望達成的目標。

當我們訂定這個架構後，使用者就可以輕易的透過 ontology 描述檔與其他的監控系統分享資料，達到 ontology 資料共享的目的。傳統的監控系統在這個部份的作法，可能採用自訂格式的文字檔或是資料庫來記錄這些訊息。其他的系統若想要讀取這些資料，就必須要事先了解自訂格式之文字檔的內容或是透過資料庫授權的方式，才有可能達到資訊分享的功能。這不僅造成管理上的困難，同時也增加程式撰寫的複雜度。

四、Ontology 設計

由於 W3C 目前採用 OWL[10]作為語意網的主要描述語言，因此，本論文也將會使用 OWL 作為撰寫 ontology 的主要標準規範。

我們將描述一個智慧型監控系統的方式，分成三個部分：裝置(Device)、事件(Event)判斷及應用(Application)邏輯，並將這三部分的資料獨立描述於程式碼之外，以期能在修改系統或在狀況發生時，即時依據現況動態的重置系統組態，減少程式需要修改及停機的時間，達到智慧型監控的目的。

因此，在我們的系統中，Ontology 描述檔主要分成三個部分：設備(Device) ontology 描述、

事件(Event) ontology 描述與應用(Application) ontology 系統邏輯描述，是由使用者自行定義的 XML 文字檔。設備描述主要是用來描述設備的基本功能及規格資料，如設備名稱、製造廠商、韌體版本等。事件描述則是用來描述此監控系統對設備發生問題時或達到某個臨界值時要如何處理或與其他設備間如何互動的描述及處理反應流程。應用系統邏輯描述則是負責描述監控系統應用情境的運作邏輯。

將 ontology 系統描述區分成這三個部分的好處是可以讓系統更加具有彈性及更容易修改系統以符合需求。舉例來說，假設有一個需要判斷室溫過高的事件，通常程式設計師會將溫度的臨界值及反應動作以程式碼的形式寫在程式中或資料庫裡，如：

```
if ($temperature > 37) {
    moveCamera(2);
    startRecording(2);
    ..... and more .....
}
```

日後要修改臨界值 37 或此事件的反應動作時，就必須要找出這段程式碼，並加以修改成新的數值。雖然說臨界溫度 37 可以將其以參數方式記錄或直接記錄在資料中，日後修改只需要更改資料庫或參數中的數值即可，也不需要更動到程式碼。但由於寫在資料庫裡欄位所代表的含義通常缺乏標準，也無法直接由資料庫的欄位其得知含義，因此我們並未使用資料庫的記錄方式來記錄，而是使用有意義的描述方式來描述這段邏輯。若採用 ontology 來描述這段邏輯時，上面事件判斷的描述方式就會變成是：

```
<event rdf:about="#室溫過高">
  <domain rdf:about="#thermometer"></domain>
  <isgreater
    rdf:about="temperature">37</isgreater>
  <action>
    <moveCamera></moveCamera>
    <recording></recording>
  </action>
</event>
```

兩者不同的地方在於：

- 日後修改臨界溫度時，可以不需要修改原來系統的程式碼，而只需要修改外在的組態檔，即可達到與修改程式碼相同的作用。

表一、感應式溫度計規格

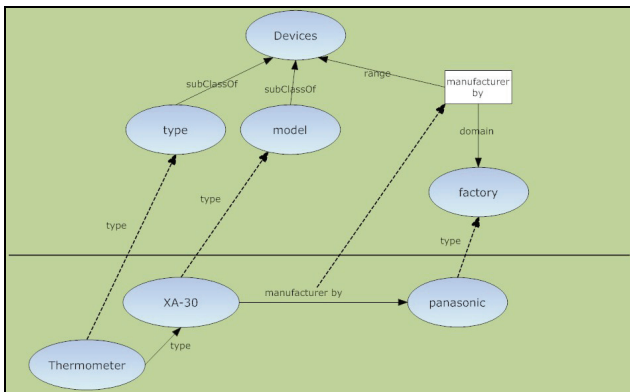
| Panasonic XA-30 數位式溫度計 | |
|------------------------|---------------|
| 型號： | XA-30 |
| 製造商： | 松下 Panasonic |
| 感應範圍： | -25°C ~ 190°C |
| 形式： | 數位感應式 |
| 感應溫度之 URI: | 依實際安裝位置 |

- 透過描述可以直覺的由字面了解意義，而不像程式碼需要經過訓練才能瞭解程式碼的涵義。
- 不需要指定特定的裝置，只需要描述事件成立的條件。可以更讓系統在啟動後依照實際的狀況彈性的決定使用的裝置。這個例子中，前述的寫法需要將使用到的鏡頭編號 2，傳入 moveCamera(2)中，以便系統決定要移動那隻鏡頭，但在 ontology 描述中卻未指定要移動那隻鏡頭，而是單純描述事件成立時需要執行的動作，等待系統初始化後再決定是那個設備。
- 若鏡頭裝置 2 損壞無法連線時，採用前面方法的程式便會發生錯誤，因為鏡頭 2 已無法連線，此時如果使用 ontology 的系統，便可以透過 ontology 的描述語言去搜尋其他可用的設備，改用替代鏡頭來執行原動作而不受鏡頭 2 損壞的影響。

由上例可知，採用 ontology 作描述的系統，會較傳統的方式來的更有彈性。以下謹就三類 ontology 文件的設計，做進一步的說明。

(一) 設備 ontology 文件

由於設備製造商在其製造的設備出廠時，並不會知道日後裝置在安裝時的相關資訊，必須要等到系統整合商實際安裝裝置後，才能得知其安裝資訊。因此，我們在設計本系統時將 device ontology 細分成出廠時的設備規格描述檔(device specification ontology)及設備安裝描述檔(device Installation ontology)，前者供製造商於設備出廠時用來描述設備的規格文件，後者供系統整合商用來描述設備實際安裝後的相關資訊，進入本系統後再組合成 device ontology 以便系統使用。



圖一、溫度計規格屬性

```

<?xml version="1.0" encoding="big5"?>
<!DOCTYPE owl [ <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" > ] >
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Class rdf:ID="TMP-001">
    <rdfs:comment>數位式溫度計 Digital Thermometer</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#Thing" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#deviceType" />
        <owl:hasValue rdf:datatype="string">thermometer</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#manufactured by" />
        <owl:hasValue rdf:datatype="string">Panasonic</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#model" />
        <owl:hasValue rdf:datatype="string">XA-30</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#temperature" />
        <owl:hasValue rdf:datatype="float">190</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#UpBound" />
        <owl:hasValue rdf:datatype="integer">190</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#LowerBound" />
        <owl:hasValue rdf:datatype="integer">-25</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
</rdf:RDF>

```

圖二、Panasonic XA-30 數位感應溫度計 ontology 規格描述檔

設備規格描述檔可讓本系統了解某項設備所在出廠時具有的功能，以便在選用設備時能夠依據使用者的需求選用符合功能的設備。舉例來說，某項溫度計設備，其出廠時規格如表一所示。此溫度計的規格屬性有：裝置類型(device type)、型號(model)、製造商(manufacturer)、感應溫度來源(temperature)、感應範圍(up-bound,

```

<event rdf:about="室溫過高">
  <domain rdf:about='thermometer'></domain>
  <isgreater rdf:about='temperature'>50</isgreater>
  <action>
    <moveCamera></moveCamera>
    <recording></recording>
  </action>
</event>

```

圖三、事件 ontology 的範例

lower-bound)等，其屬性規格可以透過 ontology 的表達方式將其關係描述如圖一所示。廠商可以根據這個規格表撰寫設備的規格描述檔(Device specification ontology)，如圖二的範例所示。

(二) 事件 ontology 文件

此文件主要用來描述監控過程中事件發生之判斷及處理的 ontology 描述檔。系統整合商可以在這個描述檔中定義所有的監控事件及反應程序，以便事件發生時作為本系統處理的依據。舉例來說，溫度計的主要功能為量測目前溫度的數值；若要用於監控，則必須要訂出一臨界數值以便判斷溫度過高或溫度過低。要用來判斷火警，則需要訂出較高的臨界值以便判斷。假設訂定量測溫度超過 50°C 為溫度過高事件，當此事件成立時有可能是該地點發生火警，管理者必須要確定該地點是否真的發生火災或是只是溫度計故障，此時若系統可以搜尋可用的鏡頭，並將鏡頭移動到該地點，透過回傳的現場影像，管理者便可以知道是否有火警發生，並決定是否要通知消防隊獲採取相關應應措施。透過 events ontology 便可以針對上述狀況加以定義描述，如圖三所示。

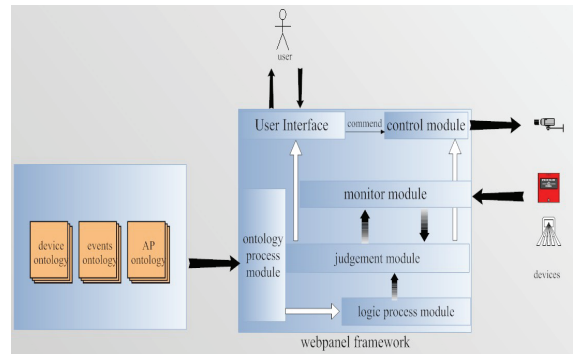
由這段事件描述檔(event ontology)可以看出系統整合商對事件的描述，描述中並不需要指定使用那項設備，只需交代事件發生後需要執行的動作，讓本系統在執行階段才決定目前有那些設備符合此事件的需求，並選用最適當的裝置來執行監控。如此做可以將整個監控系統設計的知識規則累積下來，日後系統整合商在遇到類似的狀況時，便可以重複使用這個規則，而不需要限定需使用特定設備，也可以避免不同的設計者有不同的邏輯規則設計，造成日後維護困難的狀況發生。

```

<owl:Class rdf:ID="monitorSystem">
  <rdfs:comment>火災警報監控系統</rdfs:comment>
  <area rdf:ID="#大廳">
    <device rdf:about="#大門監控鏡頭"></device>
    <device rdf:about="#大門溫度感應器">
</device>
    <event rdf:about="#室溫過高">
      <domain rdf:about="#thermometer"></domain>
      <isgreater rdf:about='temperature'>37</isgreater>
      <action>
        <moveCamera></moveCamera>
      </action>
    </event>
  </area>
</owl:Class>
</rdf:RDF>

```

圖四、應用程式 ontology



圖五、Webpanel 架構圖

(三) 應用 ontology 文件

Application ontology(AP ontology)主要是用來整合 device ontology、event ontology 再加上其他描述所組成的描述檔。系統整合商依據裝置實際安裝後得到裝置描述檔再加上需要監控的事件所定義出的各種事件，便可以完成監控系統的完整事件描述。前面有提到，由於我們並未採用 UPnP 的作法，因此系統整合商獲得設備實際安裝的資訊後，除了填寫設備安裝描述檔，還需要將此設備資訊描述於 Application ontology 中，如圖四所示。

整個 Application ontology 可以有多組 <area></area> 描述，用來表示不同的監控區域。透過 area 標籤，系統整合商可以將不同區域的監控描述都整合在一個描述檔。此外，<device rdf:about=" #大門監控鏡頭" ></device> 是系統整合商在安裝完設備後需要在應用程式描述檔中加入的描述。webpanel 會依據此描述去讀取系統整合商先前上傳的設備安裝描述檔，並獲得實際安裝資訊。<event></event> 區段表示為事件描述部分，用來描述系統在運行期間需要偵測的事件及其反應動作。

五、Webpanel 系統設計

為了要讓使用者定義的 ontology 描述檔能夠被程式轉譯並且執行，我們實作了一個名為 webpanel 的系統來達成這個目的。webpanel 的伺服器使用 ontology 做為監控系統的描述檔，因此使用者可以自行定義感應器類別及監控感應器所需要的流程。定義完成後，便可以將此

ontology 透過 webpanel 內建的上傳介面以 HTTP 通訊協定的方式上傳到 webpanel 伺服器，webpanel 中的 XML/RDF(s) 解譯模組即會開始處理使用者傳送過來的 ontology 描述檔，建立監控流程並將監控設備的硬體功能對應到系統中的功能函數，啟動使用者定義的監控流程，並對監測結果作出適當的反應。

(一) 系統架構與模組

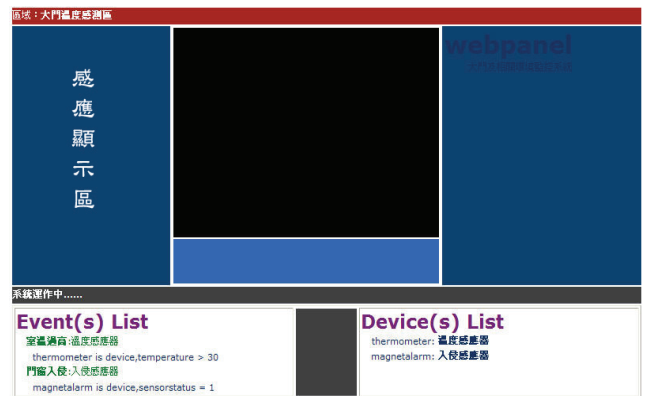
圖五為 Webpanel 監控系統的架構圖。圖左側部分便是使用 ontology 描述之裝置、判斷及應用程式邏輯的描述檔。透過 ontology process module 將這些檔案讀取並分析後，轉變成系統的設定檔及邏輯判斷依據，並由 logic process module 負責建立監控規則，交由 Judgement module 負責邏輯的判斷。Monitor module 主要負責監控裝置目前的狀態，並將偵測到的狀態交由 judgement module 做判斷。Judgement module 會依據 event ontology 來決定收到的資訊是否達到使用者設定監控的數值。若是，則依據使用者設定的 application ontology 來做出適當的反應。若有需要，則可以在 User Interface 將此訊息呈現給使用者。而使用者則可以透過 User Interface 介面來觀看目前系統各裝置的狀態或是對某裝置送出操控指令。使用者送出的指令，則透過 control module 來對裝置作控制。

監控系統採用此方式建構的好處之一，便是日後若系統組態發生異動，只需更改相關的 ontology 描述檔，系統即可自動重新載入並依據新的組態檔來重新讓系統運行。在 webpanel 系統架構中，我們將系統分成六大模組，各自負責不同的職責。

- Ontology process module:**
 主要負責讀取並處理使用者上傳的 ontology 檔案。當使用者將編寫完的 ontology 組態檔上傳後，ontology process module 會檢查上傳檔案的語法是否符合相關的規則，若不符合則會將錯誤訊息告知使用者請使用者修正。檢查正確後，此模組會將內容傳送到 login process module 中做進一步的邏輯轉換。
- Logic process module:**
 接收到格式正確的資料後，login process module 會嘗試將內容轉譯成邏輯規則，並決定要監控這些事件規則所需要使用的設備。當要決定使用某項設備時，此模組會先檢查目前環境中有那些設備是可以使用的，並從中決定最適合的設備來套用。
- Judgement module:**
 主要負責的工作有兩項：一是判斷由 monitor module 傳送過來的監控設備數值是否達到使用者設定的臨界值；二是執行事件成立時需要執行的動作。如果需要在畫面上呈現時，此模組還會將相關的監控資料及訊息遞送到 user interface，由該模組處理資料訊息顯示給使用者。
- Monitor module:**
 主要任務便是依據組態檔及 login process module 決定監控的設備，在系統運作期間持續監控相關設備當前的狀態。當此模組接收到設備傳送過來的狀態後，便會將其轉換成數值交由 judgement module 去做判斷是否要做進一步的處理。
- User Interface:**
 主要負責處理供使用者觀看的呈現介面及相關設備的訊息。本系統是採用 web-base 的介面呈現給使用者。
- Control module:**
 負責接收處理由使用者透過 user interface 傳送過來對設備所下的命令。

(二) 使用者介面

目前 webpanel 採用的是主從(client-server) 架構，在 Linux 環境下以 java 語言搭配 php 程式所寫成，以便日後能夠移植到其他平台使用。



圖六、webpanel 執行介面

表二、磁簧開關規格

| MOMRON A22LGR24A11 磁簧開關 | |
|-------------------------|-------------|
| 型號： | A22LGR24A11 |
| 製造商： | MOMRON |
| 感應器狀態： | URI |
| 形式： | 磁簧感應式開關 |

使用者介面是 Client 端的呈現介面，主要使用 html 及 javascript 來作為呈現畫面背後的語言程式。使用者可以透過此介面查看設備狀態、鏡頭影像、或控制設備、檢視報表等功能。

圖六為 webpanel 使用者端的執行的畫面。圖左上方側為感應區塊，需要偵測的事件皆會在此顯示其目前的狀態，畫面中間為監控螢幕顯示區，當使用者想要看某區域的監控螢幕時，便可以在此觀看。右方則為系統名稱及說明備註顯示區。畫面左下為監控事件列表區，顯示系統整合商設定的監控事件列表及成立條件，右下方則是目前 webpanel 於此區域所偵測到可用的裝置。透過此 web 介面，可以提供使用者檢視目前 webpanel 監控系統的各項資料及狀況，使用者也可以透過此畫面對裝置進行操控。

六、實驗結果

以下我們將以新設備如何加入以 ontology 為基礎的 web panel 伺服器作為範例，來說明使用者要如何設計 ontology 監控流程描述檔。

假設我們希望在 web panel 伺服器架構中加入用來監控大門開關的磁簧感應式開關，並想將其安裝在大門，以便監控下班後大門的進出狀

```
<?xml version="1.0" encoding="big5"?>
<!DOCTYPE owl [ <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" > ]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Class rdf:ID="GUARD">
    <rdfs:comment>Magnet Alarm</rdfs:comment>
    <rdfs:subClassOf rdf:resource="#magnetalarm" />
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#deviceType" />
        <owl:hasValue rdf:datatype="string">magnetalarm</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#manufactured by" />
        <owl:hasValue rdf:datatype="string">MOHRON</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#model" />
        <owl:hasValue rdf:datatype="string">A22LGR24A11</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
    <rdfs:subClassOf>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#sensorstatus" />
        <owl:hasValue rdf:datatype="float">1</owl:hasValue>
      </owl:Restriction>
    </rdfs:subClassOf>
  </owl:Class>
</rdf:RDF>
```

圖七、A22LGR24A11 磁簧開關規格描述檔

```
<?xml version="1.0" encoding="big5"?>
<!DOCTYPE owl [ <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" > ]>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
  <owl:Class rdf:ID="磁簧開關">
    <rdfs:comment>磁簧感應開關</rdfs:comment>
    <rdfs:subClassOf rdf:about="#GUARD" />
    <sensorstatus>http://192.168.0.7/webpanel/monitor.php?param=1</sensorstatus>
  </owl:Class>
</rdf:RDF>
```

圖九、磁簧開關設備安裝描述 ontology 檔

```
<event rdf:about="#門窗入侵">
  <domain rdf:about="#device" />
  <range rdf:about="#magnetalarm" />
  <equal rdf:about="sensorstatus">1</equal>
</event>
```

圖十、門窗入侵事件 ontology 描述檔

```
- <actions>
- <moveCamera rdf:about="#移動鏡頭">
  <domain rdf:about="#device" />
  <range rdf:about="#camera" />
  <preset rdf:about="大門" />
</moveCamera>
</actions>
```

圖十一、移動鏡頭 ontology 描述



圖八、製造商上傳設備規格描述 ontology 檔畫面

況。一般製造廠商對於此磁簧開關的規格描述多如表二所示，我們假設製造商可依照此磁簧開關的規格，以 ontology 的語法編寫如圖七的設備規格描述檔。

設計完規格描述檔後，廠商就可以透過 webpanel 提供，如圖八所示的上傳介面將此描述檔至系統中。廠商上傳完畢後，webpanel 便能透過此描述檔將磁簧開關的功能及規格轉成 webpanel 所能了解的描述格式。同時由製造商的 ontology 規格檔中使用者可以了解此設備的相關資訊以做為選用與否的參考。

系統整合商於實際環境設置完磁簧開關後，便可依據前述製造商提供的設備規格描述檔將安裝後的相關資訊編寫成設備安裝描述檔，如圖九所示。

系統整合商編寫完成設備安裝描述 ontology 檔後，同樣的也需要透過上傳介面，將此檔案上傳至 webpanel 系統。若描述檔正確且裝置存在，webpanel 會對目前在線上的節點發出訊息，通知有新的裝置註冊。此時，所有節點會重新載入組態檔並將此裝置納入可用裝置。在組態重組的過程中，webpanel 會根據使用者設定的系統檔來檢測新裝置是否比原來的裝置適合使用者需求；若是，則系統會優先採用新裝置來取代原有的裝置，並將原有的裝置作為備援裝置。

在 webpanel 系統獲得相關設備的實際安裝資訊後，使用者便可以在 application ontology 中去設定 webpanel 應用系統的事件。這裡假設系統整合商希望 webpanel 系統能夠在下班後偵測大門開闔的狀況，因此針對此需求建立了大門入侵的事件，如圖十所示。此描述檔表示，當磁簧開關的 sensor status 偵測到數值為 1 時，表示門窗被打開。除了設定偵測狀態外，尚需搭配事件發生時的反應動作，這樣整個事件反映才會完整。假設系統整合商希望發生上述大門入侵事件時，能夠將攝影機移動到大門，並啟動錄影，則可以將此動作以 ontology 描述如圖十一所示。藉

```

- <owl:Class rdf:ID="webpanel">
  <rdfs:comment>大門及相關環境監控系統</rdfs:comment>
- <area rdf:ID="#大門溫度感測區">
  <device rdf:about="#溫度感應器" />
  <device rdf:about="#入侵感應器" />
- <event rdf:about="#室溫過高">
  <domain rdf:about="#device" />
  <range rdf:about="#thermometer" />
  <isgreater rdf:about="temperature">30</isgreater>
</event>

```

圖十二、手動加入磁簧開關設備描述

```

<owl:Class rdf:ID="GUARD"> 設備規格描述檔
  <rdfs:comment>Magnet Alarm</rdfs:comment>
  <rdfs:subClassOf rdf:resource="#magnetalarm" />
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#deviceType" />
      <owl:hasValue rdf:datatype="string">magnetalarm</owl:hasValue>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>

設備安裝描述檔
<owl:Class rdf:ID="磁簧開關">
  <rdfs:comment>磁簧感應器</rdfs:comment>
  <rdfs:subClassOf rdf:about="#GUARD" />
  <sensorstatus>http://192.168.0.7/webpanel/monitor.php?param=1&mp=2</sensorstatus>
</owl:Class>
</rdf:RDF>

```

圖十三、磁簧開關參照圖

由上面的<action> ... </action>區段描述，便可以設計各種反應動作，並與事件做一結合，以完成整個事件的監控流程設計。

將各種監控事件透過上述之設計流程，便可以組織成一個完整的監控系統邏輯，也就是webpanel的應用系統描述檔。由於我們並未使用UPnP技術，所以必須要將區域中安裝的設備以手動的方式加入到應用系統描述檔中，以便webpanel認知此區域所使用到的設備。因此，我們必須要手動的將磁簧開關設備描述加入到應用系統描述檔中，如圖十二紅色框框處所示。

在系統整合商加入上面的描述後，Logic process module 會根據"#入侵感應器"這個ID去搜尋設備安裝描述檔，以找出系統整合商先前上傳之設備安裝描述檔，並透過解析其內容可以得知其為型號 GUARD 之磁簧開關，再透過 GUARD 關鍵字去設備規格描述檔中將製造商上傳關於該設備之規格讀取出來，同時結合設備安裝描述檔之安裝資訊，以完成整個磁簧開關之描述。其參照方式可見圖十三所示。

設計完成應用系統描述檔後，便可以開始透過webpanel的web介面進行監控程序。圖十四便是webpanel使用者端的執行的畫面。圖左上



圖十四、webpanel 執行介面

方側為感應區塊，需要偵測的事件皆會在此顯示其目前的狀態，畫面中間為監控螢幕顯示區，當使用者想要看某區域的監控螢幕時，便可以在此觀看。右方則為系統名稱及說明備註顯示區。畫面左下為監控事件列表區，顯示系統整合商設定的監控事件列表及成立條件，右下方則是目前webpanel於此區域所偵測到可用的裝置。透過此web介面，可以提供使用者檢視目前webpanel監控系統的各項資料及狀況，使用者也可以透過此畫面對裝置進行操控。

若每個監控伺服器都能夠接受以ontology為基礎的設備描述檔，則要讓監控伺服器能自動查詢彼此的設備及功能，將會是一件容易被達成的事。因為ontology原本的設計就是為了要讓資訊能夠輕易的在不同的系統中交換。

七、結論及未來研究

本論文展示了如何透過ontology描述監控設備的功能以及監控的流程，並與webpanel結合，讓監控系統的設計流程簡單化。如此使用者只需專注在設備ontology描述檔的設計，而不需要管程式設計師是如何設計他們的程式。日後設備或功能有變更也只需要更動ontology即可完成系統流程的重設。

Ontology不是webpanel系統最先使用，但webpanel系統卻希望能夠將ontology便利可重複使用的觀念引導進入監控領域中，讓封閉型的監控領域能夠與更容易的與其他領域(如smart home等)相結合，為人類的生活帶來更大的便

利。

由於 ontology 以及 OWL 的研究仍然持續進行，因此未來不排除會更改 ontology 的設計，以符合標準的語法。webpanel 與 user view 的連結部份，目前是使用 javascript + DOM 的方式取得 webpanel 系統目前的狀態，每隔一段時間由 client 端主動向 server 端取得感應器的目前狀態，然後再解析訊息顯示在網頁上。本系統目前亦無法由 server 端直接 push 資訊到 client 端；未來可能會針對此點作改進，改由 java applet 的方式來被動接收訊息，server 則在有警訊發生時，才主動通知 client 端顯示相關訊息。這種做法將可以有效的減輕 server 一直接受 client 查詢訊息的負擔。

此外，如何讓使用者能夠輕易的設計出符合規範的 ontology 描述檔對 webpanel 來說也是一項挑戰。目前使用者是採用手動 key-in 的方式來建立 ontology 描檔，不僅非常耗時間同時也容易出錯，未來也會針對這點，開發客製化的程式介面，讓使用者能容易輸入這些描述檔的內容。

八、參考文獻

- [1] 陳朝揚、楊耀榮與林宜翰，”以 ontology 為基礎的資訊抽取系統之研究”，*創新研發學刊*，台南，pp. 33-39，民國 85 年 11 月。
- [2] M. Araki and A. Miyajima, “Application of Ubiquitous Web Technologies to Home Information Appliances,” in *Proc. of W3C Ubiquitous Web Workshop 2006*, Tokyo, Japan, March 2006.
- [3] T. Berners-Lee, Semantic Web Road map. <http://www.w3.org/DesignIssues/Semantic.html> (last modified Oct. 14, 1998).
- [4] M. Crubezy, M. O’Connor, Z. Pincus, and M.A. Musen, “Ontology-Centered Syndromic Surveillance for Bioterrorism,” *IEEE Intelligent Systems*, vol. 20, no. 5, pp. 26-35, Sept/Oct, 2005.
- [5] N. Gibbins, S. Harris and N. Shadbolt “Agent-based Semantic Web Services,” in *Proc. of the 12th World Wide Web Conference*, Budapest, Hungary, 2003.
- [6] C.P. Hsu , “E 世代—家庭自動化系統之設計,” *Workshop on the 21st Century Digital Life and Internet Technologies*, Tainan, Taiwan, 2001.
- [7] Y.-H. Jiang, W.H. Chao, C.C. Chiang, ”Internet Application for Supervisory Control System of Appliances,” in *Proc. of Workshop on the 21th Century Digital Life and Internet Technologies*, pp. 108, Tainan, Taiwan, 2001.
- [8] P. Kogut, W. Holmes, “AeroDAML: Applying information extraction to generate DAML annotations from web pages,” in *Proc. of First International Conference on Knowledge Capture (K-CAP’01)*, 2001.
- [9] K. Moller and S. Decker, “Harvesting Desktop Data for Semantic Blogging”, .in *Proc. of ISWC 2005 Workshop*, 2005.
- [10] OWL Web Ontology Language (OWL), <http://www.w3.org/2004/OWL/>, 2004.