s.k. Y. Bhg Dato' Profesor Muhammad Idiris Saleh
Timbalan Naib Canselor
[Penyelidikan & Inovasi]

Prof. Madya Bahari Belaton
Pemangku Dekan Penyelidikan
Pelantar Teknologi Maklumat & Komunikasi
Pejabat Pelantar Penyelidikan

Profesor Rosni Abdullah
Dekan
Pusat Pengajian Sains Komputer

Prof. Madya Azman Samsudin
Timbalan Dekan (Pengajian Siswazah & Penyelidikan)
Pusat Pengajian Sains Komputer

Prof. Madya Tang Enya Kong
Penyelidik Bersama Projek
Pusat Pengajian Sains Komputer

Rohani Bakar
Pegawai Sains
Pelantar Teknologi Maklumat & Komunikasi
Pejabat Pelantar Penyelidikan

Puan Sofiah Hashim
Ketua Pustakawan
Perpustakaan Hamzah Sendut 1

Disampaikan satu salinan
laporan akhir projek untuk
simpanan Perpustakaan

Puan Ansuya a/p Narhari
Penolong Bendahari
Unit Kumpulan Wang Penyelidikan
Jabatan Bendahari

Sila ambil tindakan menutup
akaun projek pada **31 Disember 2007**
dan sila kemukakan satu salinan
kewangan terakhir ke pejabat
(RCMO)

CMI/yu…laporanakhirinduk

UNIVERSITI SAINS MALAYSIA

LAPORAN AKHIR PROJEK PENYELIDIKAN JANGKA *PENDEK*

FINAL REPORT OF SHORT TERM RESEARCH PROJECT

Sila kemukakan laporan akhir ini melalui Jawatankuasa Penyelidikan di Pusat Pengajian dan Dekan/Pengarah/Ketua Jabatan kepada Pejabat Pelantar Penyelidikan

1. **Nama Ketua Penyelidik:** **Gian Chand Sodhy**
Name of Research Leader

□ Profesor Madya/ *Assoc. Prof.*　□ Dr./ *Dr.*　☑ Encik/Puan/Cik *Mr/Mrs/Ms*

2. **Pusat Tanggungjawab (PTJ):** **Pusat Pengajian Sains Komputer**
School/Department

3. **Nama Penyelidik Bersama:** **Prof. Madya Dr. Tang Enya Kong**
Name of Co-Researcher

4. **Tajuk Projek:** **Information Integration and Semantic Interoperability Support in Ontologies**
Title of Project

5. **Ringkasan Penilaian**/*Summary of Assessment*:

	Tidak Mencukupi *Inadequate*		Boleh Diterima *Acceptable*	Sangat Baik *Very Good*	
	1	2	3	4	5
i) **Pencapaian objektif projek:** *Achievement of project objectives*	□	□	□	☑	□
ii) **Kualiti output:** *Quality of outputs*	□	□	☑	□	□
iii) **Kualiti impak:** *Quality of impacts*	□	□	□	☑	□
iv) **Pemindahan teknologi/potensi pengkomersialan:** *Technology transfer/commercialization potential*	□	□	☑	□	□
v) **Kualiti dan usahasama :** *Quality and intensity of collaboration*	□	□	□	☑	□
vi) **Penilaian kepentingan secara keseluruhan:** *Overall assessment of benefits*	□	□	☑	□	□

6. Abstrak Penyelidikan

(Perlu disediakan di antara 100 - 200 perkataan di dalam **Bahasa Malaysia dan juga Bahasa Inggeris**. Abstrak ini akan dimuatkan dalam Laporan Tahunan Bahagian Penyelidikan & Inovasi sebagai satu cara untuk menyampaikan dapatan projek tuan/puan kepada pihak Universiti & masyarakat luar).

Abstract of Research
(An abstract of between 100 and 200 words must be prepared in Bahasa Malaysia and in English). This abstract will be included in the Annual Report of the Research and Innovation Section at a later date as a means of presenting the project findings of the researcher/s to the University and the community at large)

- Sila lihat lampiran -

7. Sila sediakan laporan teknikal lengkap yang menerangkan keseluruhan projek ini.

[Sila gunakan kertas berasingan]
Applicants are required to prepare a Comprehensive Technical Report explaining the project. (This report must be appended separately)

- Sila lihat kertas-kertas kerja yang dilampirkan untuk butiran teknikal projek ini -

Senaraikan kata kunci yang mencerminkan penyelidikan anda:
List the key words that reflect your research:

Bahasa Malaysia	*Bahasa Inggeris*
Maklumat	Information
Integrasi	Integration
Semantik	Semantic
Perwakilan ontologi	Ontology representation

8. Output dan Faedah Projek

Output and Benefits of Project

(a) * **Penerbitan Jurnal**
Publication of Journals
(Sila nyatakan jenis, tajuk, pengarang/editor, tahun terbitan dan di mana telah diterbit/diserahkan)
(State type, title, author/editor, publication year and where it has been published/submitted)

Ong, S.K. & Tang, E.K. (2005) Service Description and Composition Using Ontologies. In Proceedings of the 23rd IASTED International Conference on Artificial Intelligent and Applications (AIA2005), Innsbruck, Austria

Ong, S.K. & Tang, E.K. (2005) Conceptual Modeling and Reasoning Using Ontology. In Proceedings of the 1st National Computer Sciences Postgraduate Colloquium (NaCSPC'05), Penang, Malaysia

Ong, S.K. & Tang, E.K. (2005) Creating and Organizing Semantic Web Services. In Proceedings of the MMU International Symposium on Information and Communications Technologies (M2USIC2005), KL, Malaysia

Ong, S.K. (2006) Semantic Web Services Organization using Ontology, Master Thesis, School Of Computer Sciences, Universiti Sains Malaysia

(b) **Faedah-faedah lain seperti perkembangan produk, pengkomersialan produk/pendaftaran paten atau impak kepada dasar dan masyarakat.**
State other benefits such as product development, product commercialisation/patent registration or impact on source and society.

Teknologi ontologi ini semakin luas digunakan oleh syarikat-syarikat pembekal dan peminta maklumat untuk mengintegrasikan data dari pelbagai sumber Web dalam internet.

* Sila berikan salinan/*Kindly provide copies*

(c) **Latihan Sumber Manusia**
Training in Human Resources

i) Pelajar Sarjana:
Graduates Students
(Perincikan nama, ijazah dan status)
(Provide names, degrees and status)

ONG SIEW KIN (Cik) – Pelajar siswazah, M.Sc.

ii) Lain-lain:
Others

9. **Peralatan yang Telah Dibeli:**
Equipment that has been purchased

- Tiada -

Tandatangan Penyelidik
Signature of Researcher

19/9/07

Tarikh
Date

Komen Jawatankuasa Penyelidikan Pusat Pengajian/Pusat
Comments by the Research Committees of Schools/Centres

The research is looking into the issue of information integration by using ontology representation. This work has a high impact to UTMK since a few projects in UTMK are also relying on the usage of ontology. Few findings had been translated to conference papers, with one master students being trained in the research area.

Assoc. Prof. Dr. Azman Samsudin
Deputy Dean
Postgraduate Studies and Research
School of Computer Sciences
Universiti Sains Malaysia
11800 USM Penang

TANDATANGAN PENGERUSI
JAWATANKUASA PENYELIDIKAN
PUSAT PENGAJIAN/PUSAT
Signature of Chairman
[Research Committee of School/Centre]

Oct 1, 2007
Tarikh
Date

6. Abstrak Penyelidikan

Jumlah maklumat yang dapat diperolehi dalam talian adalah banyak. Salah satu usaha dalam mengintegrasikan maklumat dari segi sintaks and semantik adalah dengan perwakilan ontologi. Ontologi mentakrifkan maklumat semantik secara formal supaya dapat diproses oleh komputer. Kami telah merekabentuk satu model organisasi yang terdiri daripada 4 lapisan ontologi untuk mewakili maklumat dalam perkhidmatan dalam bidang pelancongan. Empat lapisan ini ialah *upper service ontology*, *domain service category ontology*, *domain ontology* dan *state ontology*. *Upper service ontology* yang dicadangkan adalah umum dan abstrak supaya dapat mewakili kebanyakan perkhidmatan web. Tiga lapisan ontologi yang lain adalah spesifik kepada bidang tertentu. Dalam projek ini, kami telah mengenalpasti kaedah-kaedah dalam pembangunan ontologi. Kaedah kami mengutamakan penggunaan sumber yang sedia ada seperti aplikasi web, skema UNSPSC, spesifikasi OTA dan WORDNET dalam pembangunaan ontologi. Ontologi yang telah dihasilkan digunakan dalam mewakili maklumat dalam aplikasi penerbangan dalam talian. Pada masa kini, ontologi kami mengandungi 845 konsep (rujuk lampiran).

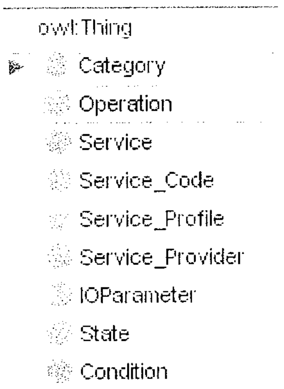
Abstract of Research

The amount of information available online is enormous. One of the efforts in integrating information syntactically and semantically is using ontology representation. Ontology defines formal semantics for information and allows processing by computer. We have designed an organization model which consists of four layers of ontology for representing services in tourism domain. They are upper service ontology, domain service category ontology, domain ontology and state ontology.The suggested upper service ontology is common and abstact enough to describe all web services while the three remaining ontologies are domain dependent. In the project, we have identified the methodologies in creating each type of ontologies. Our method focuses on using exsiting resources like web applications, UNSPSC schemas, OTA specification and WORDNET in creating the ontologies. The resulted ontologies were used in annotating online flight application. Currently, there are 845 concepts captured in the ontologies (see attachment).

Lampiran / Attachment

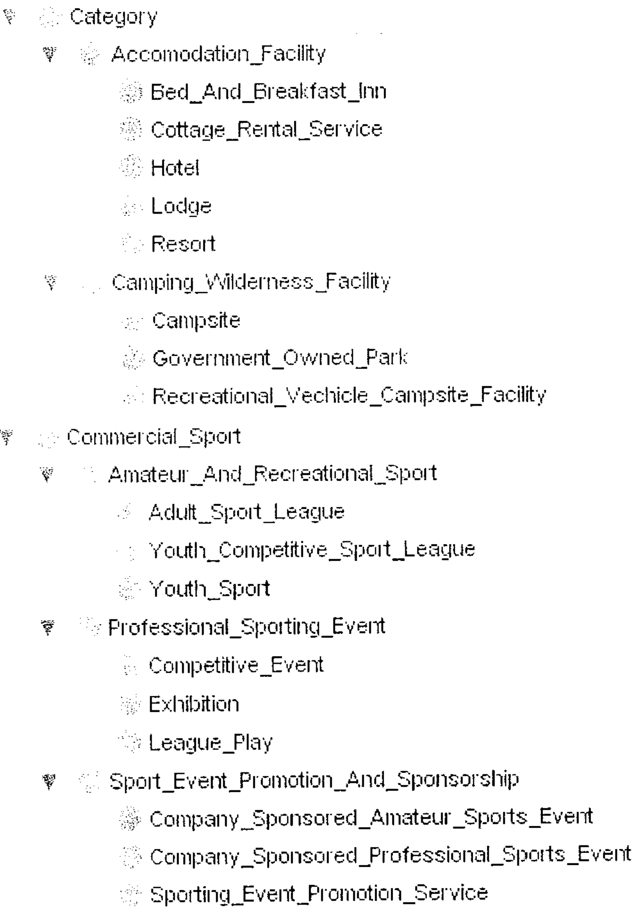
Concepts of Ontologies

re A1 - figure A5 show the ontologies created. The screen shots which illustrating the class hierarchies e taken from Protégé 2000 editor software.

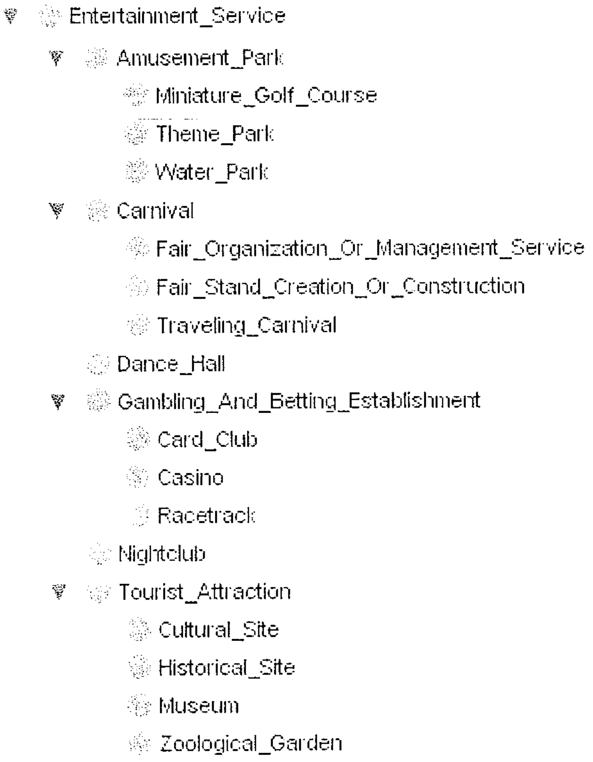


- owl:Thing
  - Category
  - Operation
  - Service
  - Service_Code
  - Service_Profile
  - Service_Provider
  - IOParameter
  - State
  - Condition

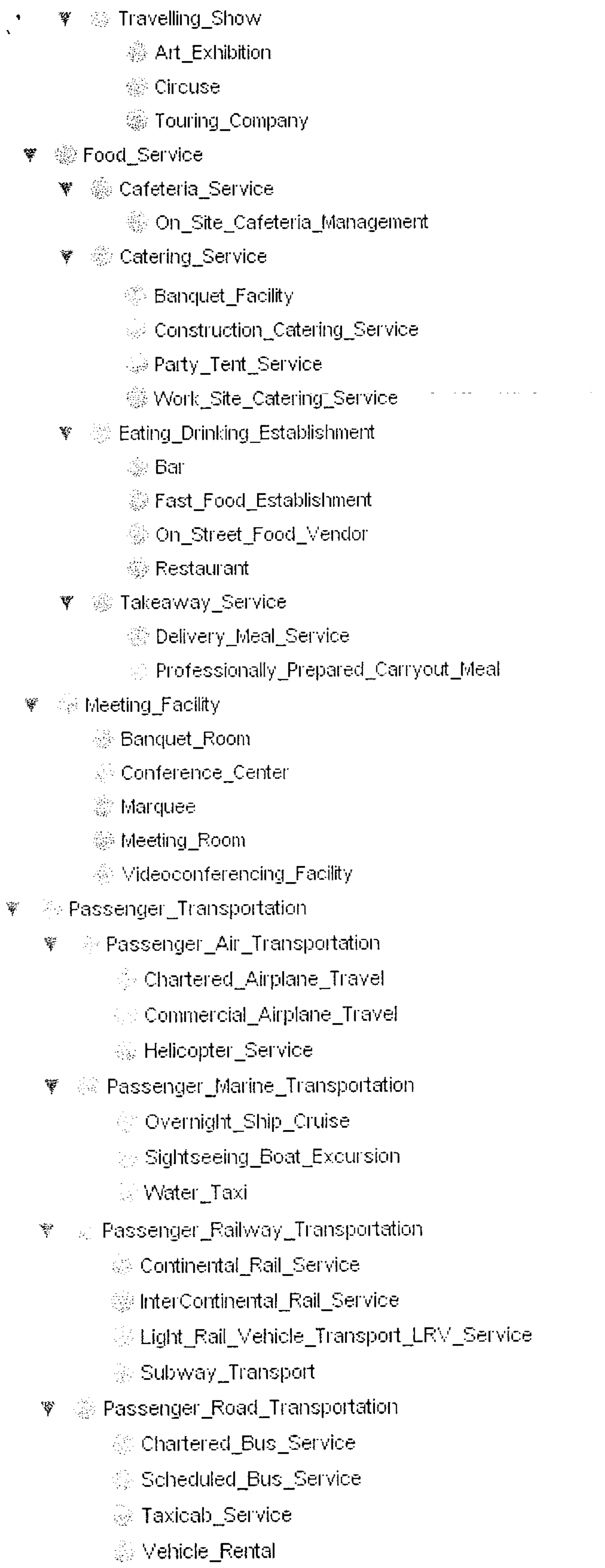
Figure A1 Upper service ontology



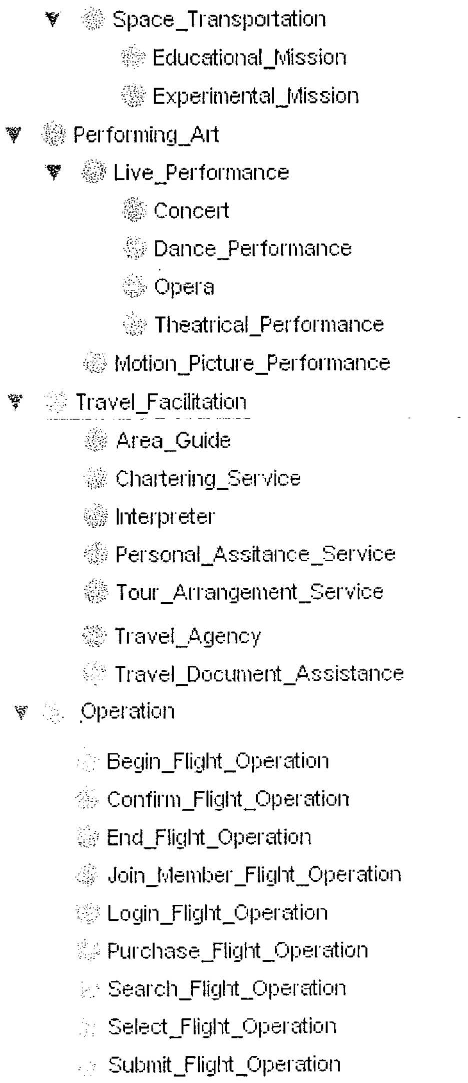
- Category
  - Accomodation_Facility
    - Bed_And_Breakfast_Inn
    - Cottage_Rental_Service
    - Hotel
    - Lodge
    - Resort
  - Camping_Wilderness_Facility
    - Campsite
    - Government_Owned_Park
    - Recreational_Vechicle_Campsite_Facility
- Commercial_Sport
  - Amateur_And_Recreational_Sport
    - Adult_Sport_League
    - Youth_Competitive_Sport_League
    - Youth_Sport
  - Professional_Sporting_Event
    - Competitive_Event
    - Exhibition
    - League_Play
  - Sport_Event_Promotion_And_Sponsorship
    - Company_Sponsored_Amateur_Sports_Event
    - Company_Sponsored_Professional_Sports_Event
    - Sporting_Event_Promotion_Service



- Entertainment_Service
  - Amusement_Park
    - Miniature_Golf_Course
    - Theme_Park
    - Water_Park
  - Carnival
    - Fair_Organization_Or_Management_Service
    - Fair_Stand_Creation_Or_Construction
    - Traveling_Carnival
  - Dance_Hall
  - Gambling_And_Betting_Establishment
    - Card_Club
    - Casino
    - Racetrack
  - Nightclub
  - Tourist_Attraction
    - Cultural_Site
    - Historical_Site
    - Museum
    - Zoological_Garden

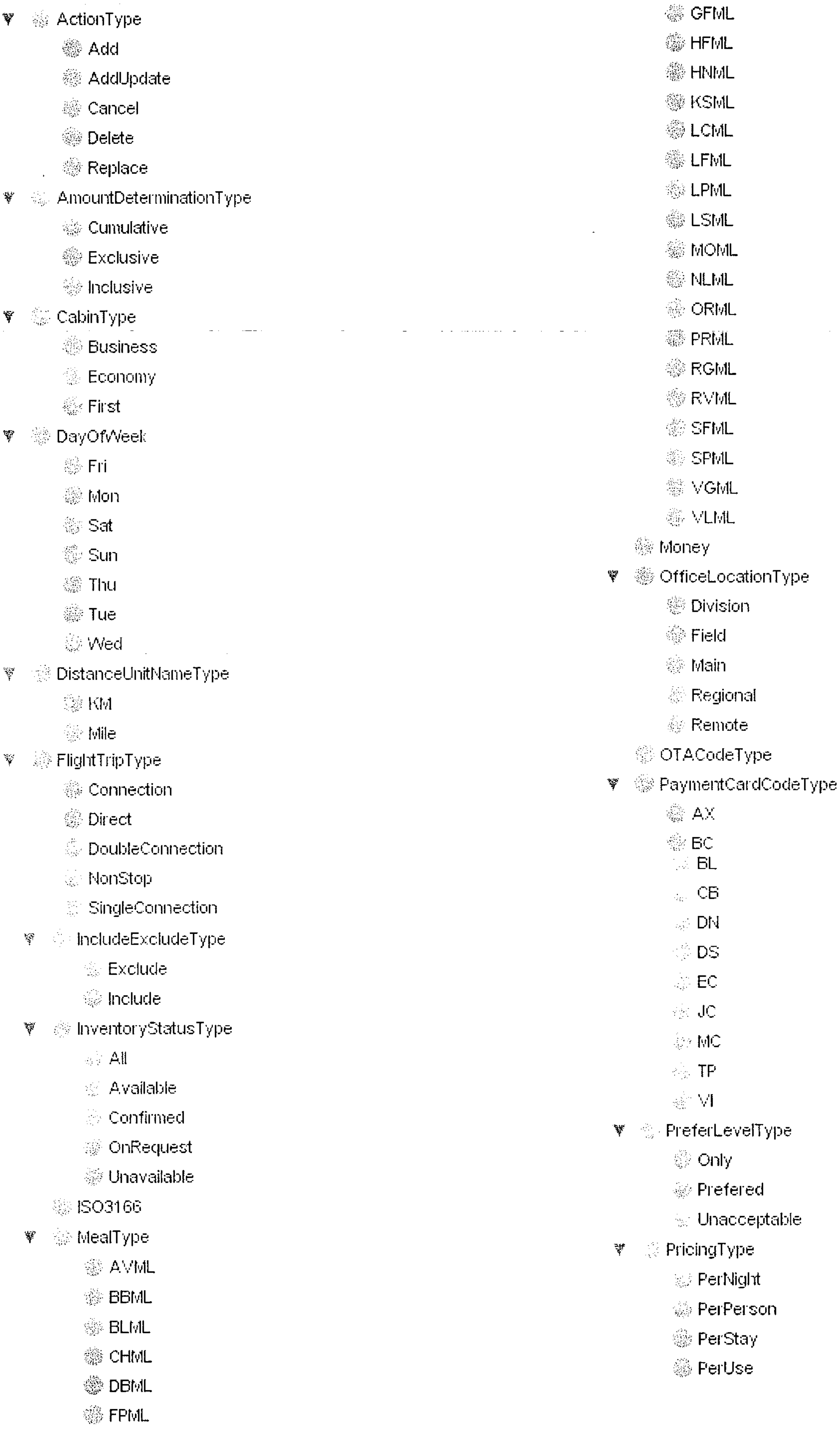


- Travelling_Show
    - Art_Exhibition
    - Circuse
    - Touring_Company
- Food_Service
    - Cafeteria_Service
        - On_Site_Cafeteria_Management
    - Catering_Service
        - Banquet_Facility
        - Construction_Catering_Service
        - Party_Tent_Service
        - Work_Site_Catering_Service
    - Eating_Drinking_Establishment
        - Bar
        - Fast_Food_Establishment
        - On_Street_Food_Vendor
        - Restaurant
    - Takeaway_Service
        - Delivery_Meal_Service
        - Professionally_Prepared_Carryout_Meal
- Meeting_Facility
    - Banquet_Room
    - Conference_Center
    - Marquee
    - Meeting_Room
    - Videoconferencing_Facility
- Passenger_Transportation
    - Passenger_Air_Transportation
        - Chartered_Airplane_Travel
        - Commercial_Airplane_Travel
        - Helicopter_Service
    - Passenger_Marine_Transportation
        - Overnight_Ship_Cruise
        - Sightseeing_Boat_Excursion
        - Water_Taxi
    - Passenger_Railway_Transportation
        - Continental_Rail_Service
        - InterContinental_Rail_Service
        - Light_Rail_Vehicle_Transport_LRV_Service
        - Subway_Transport
    - Passenger_Road_Transportation
        - Chartered_Bus_Service
        - Scheduled_Bus_Service
        - Taxicab_Service
        - Vehicle_Rental

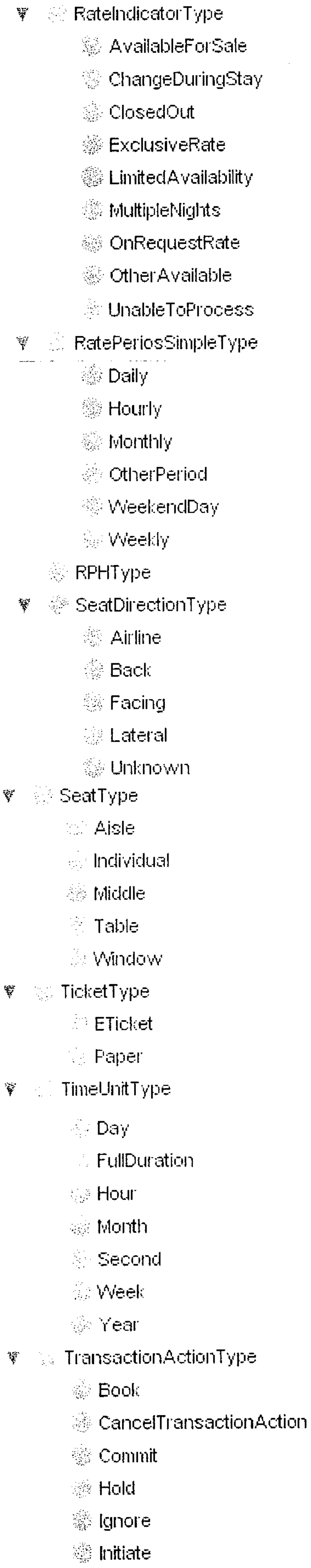


- Space_Transportation
    - Educational_Mission
    - Experimental_Mission
- Performing_Art
    - Live_Performance
        - Concert
        - Dance_Performance
        - Opera
        - Theatrical_Performance
    - Motion_Picture_Performance
- Travel_Facilitation
    - Area_Guide
    - Chartering_Service
    - Interpreter
    - Personal_Assitance_Service
    - Tour_Arrangement_Service
    - Travel_Agency
    - Travel_Document_Assistance
- Operation
    - Begin_Flight_Operation
    - Confirm_Flight_Operation
    - End_Flight_Operation
    - Join_Member_Flight_Operation
    - Login_Flight_Operation
    - Purchase_Flight_Operation
    - Search_Flight_Operation
    - Select_Flight_Operation
    - Submit_Flight_Operation

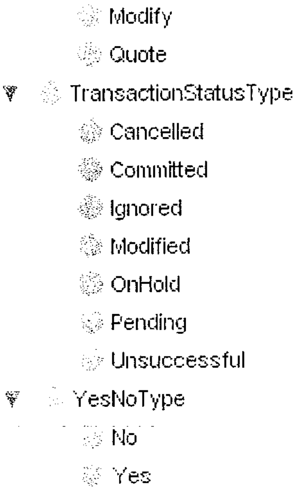
Figure A2 Travel service category ontology



- ActionType
  - Add
  - AddUpdate
  - Cancel
  - Delete
  - Replace
- AmountDeterminationType
  - Cumulative
  - Exclusive
  - Inclusive
- CabinType
  - Business
  - Economy
  - First
- DayOfWeek
  - Fri
  - Mon
  - Sat
  - Sun
  - Thu
  - Tue
  - Wed
- DistanceUnitNameType
  - KM
  - Mile
- FlightTripType
  - Connection
  - Direct
  - DoubleConnection
  - NonStop
  - SingleConnection
- IncludeExcludeType
  - Exclude
  - Include
- InventoryStatusType
  - All
  - Available
  - Confirmed
  - OnRequest
  - Unavailable
- ISO3166
- MealType
  - AVML
  - BBML
  - BLML
  - CHML
  - DBML
  - FPML
  - GFML
  - HFML
  - HNML
  - KSML
  - LCML
  - LFML
  - LPML
  - LSML
  - MOML
  - NLML
  - ORML
  - PRML
  - RGML
  - RVML
  - SFML
  - SPML
  - VGML
  - VLML
- Money
- OfficeLocationType
  - Division
  - Field
  - Main
  - Regional
  - Remote
- OTACodeType
- PaymentCardCodeType
  - AX
  - BC
  - BL
  - CB
  - DN
  - DS
  - EC
  - JC
  - MC
  - TP
  - VI
- PreferLevelType
  - Only
  - Prefered
  - Unacceptable
- PricingType
  - PerNight
  - PerPerson
  - PerStay
  - PerUse

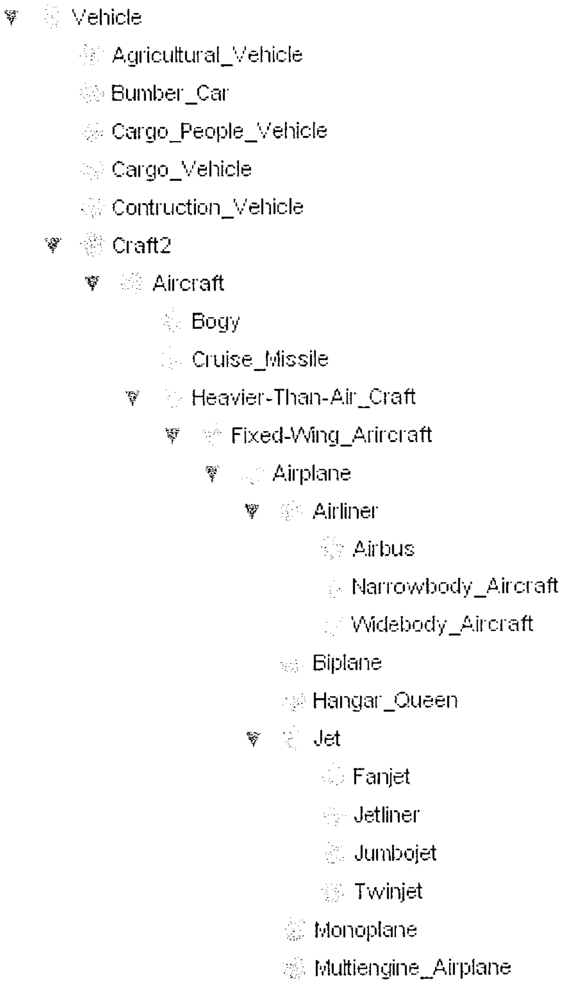


- RateIndicatorType
  - AvailableForSale
  - ChangeDuringStay
  - ClosedOut
  - ExclusiveRate
  - LimitedAvailability
  - MultipleNights
  - OnRequestRate
  - OtherAvailable
  - UnableToProcess
- RatePeriosSimpleType
  - Daily
  - Hourly
  - Monthly
  - OtherPeriod
  - WeekendDay
  - Weekly
- RPHType
- SeatDirectionType
  - Airline
  - Back
  - Facing
  - Lateral
  - Unknown
- SeatType
  - Aisle
  - Individual
  - Middle
  - Table
  - Window
- TicketType
  - ETicket
  - Paper
- TimeUnitType
  - Day
  - FullDuration
  - Hour
  - Month
  - Second
  - Week
  - Year
- TransactionActionType
  - Book
  - CancelTransactionAction
  - Commit
  - Hold
  - Ignore
  - Initiate

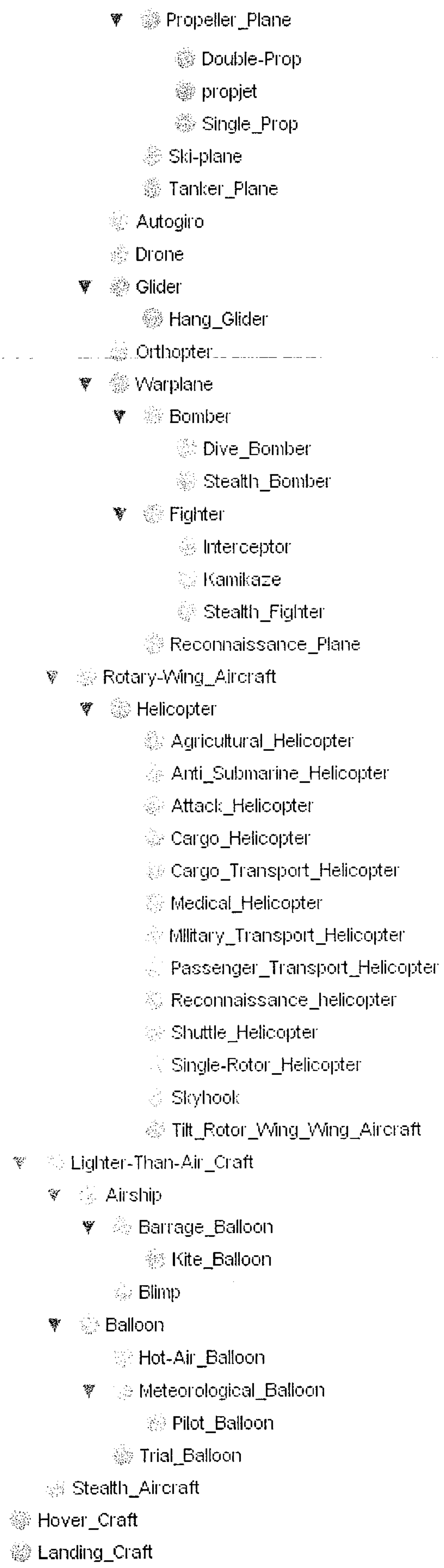


  - Modify
  - Quote
- TransactionStatusType
  - Cancelled
  - Committed
  - Ignored
  - Modified
  - OnHold
  - Pending
  - Unsuccessful
- YesNoType
  - No
  - Yes

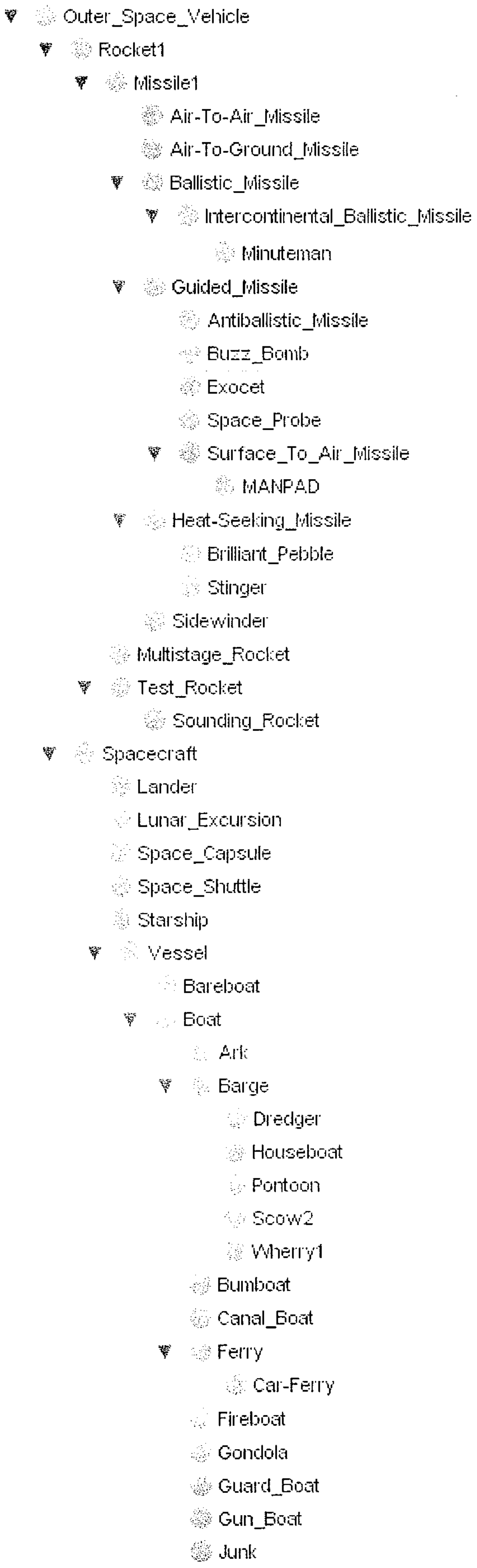
Figure A3 Flight business data ontology



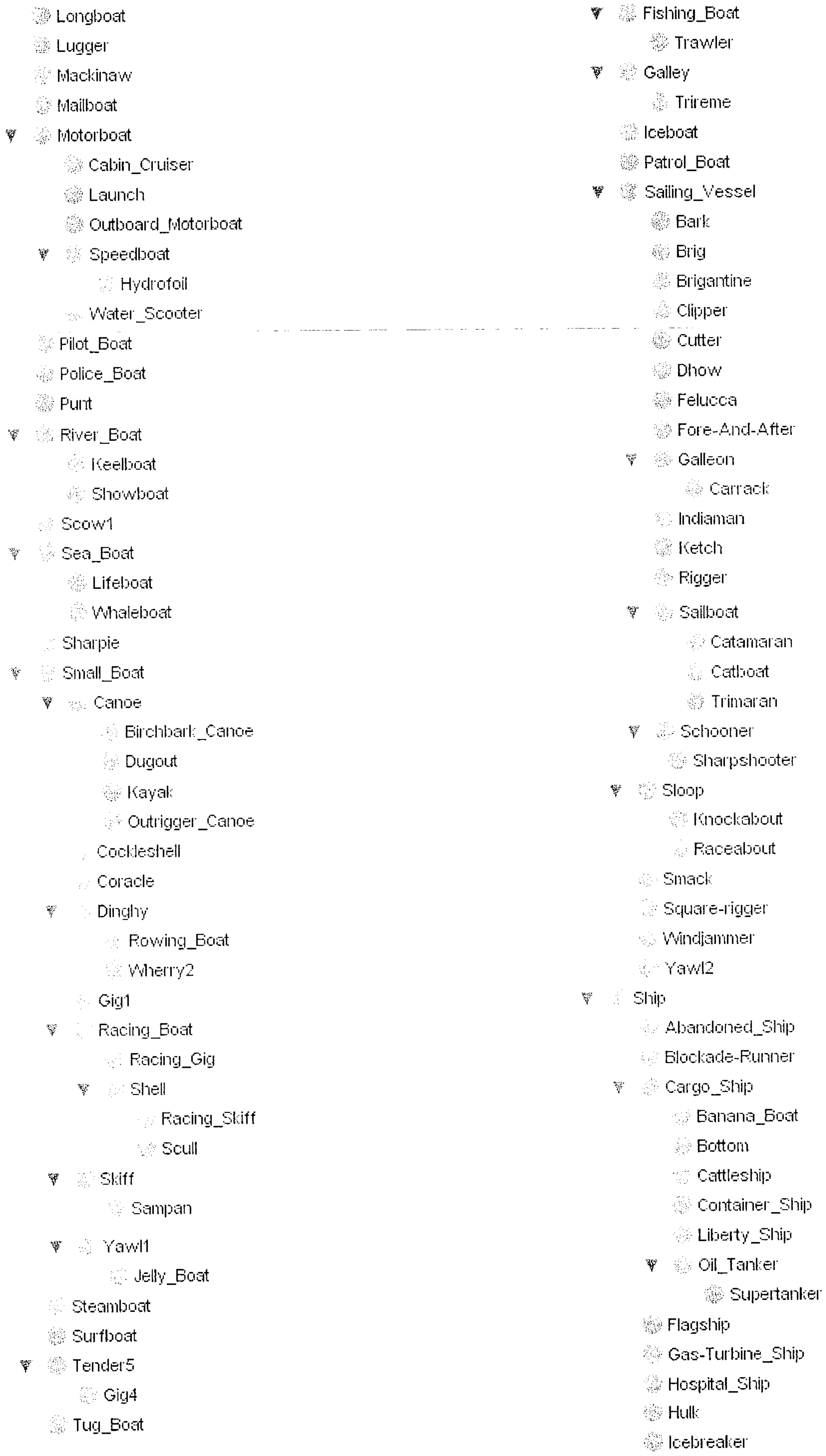
- Vehicle
  - Agricultural_Vehicle
  - Bumber_Car
  - Cargo_People_Vehicle
  - Cargo_Vehicle
  - Contruction_Vehicle
  - Craft2
    - Aircraft
      - Bogy
      - Cruise_Missile
      - Heavier-Than-Air_Craft
        - Fixed-Wing_Arircraft
          - Airplane
            - Airliner
              - Airbus
              - Narrowbody_Aircraft
              - Widebody_Aircraft
            - Biplane
            - Hangar_Queen
            - Jet
              - Fanjet
              - Jetliner
              - Jumbojet
              - Twinjet
            - Monoplane
            - Multiengine_Airplane



```
                Propeller_Plane
                    Double-Prop
                    propjet
                    Single_Prop
                Ski-plane
                Tanker_Plane
            Autogiro
            Drone
            Glider
                Hang_Glider
            Orthopter
            Warplane
                Bomber
                    Dive_Bomber
                    Stealth_Bomber
                Fighter
                    Interceptor
                    Kamikaze
                    Stealth_Fighter
                Reconnaissance_Plane
        Rotary-Wing_Aircraft
            Helicopter
                Agricultural_Helicopter
                Anti_Submarine_Helicopter
                Attack_Helicopter
                Cargo_Helicopter
                Cargo_Transport_Helicopter
                Medical_Helicopter
                Military_Transport_Helicopter
                Passenger_Transport_Helicopter
                Reconnaissance_helicopter
                Shuttle_Helicopter
                Single-Rotor_Helicopter
                Skyhook
                Tilt_Rotor_Wing_Wing_Aircraft
    Lighter-Than-Air_Craft
        Airship
            Barrage_Balloon
                Kite_Balloon
            Blimp
        Balloon
            Hot-Air_Balloon
            Meteorological_Balloon
                Pilot_Balloon
            Trial_Balloon
    Stealth_Aircraft
Hover_Craft
Landing_Craft
```

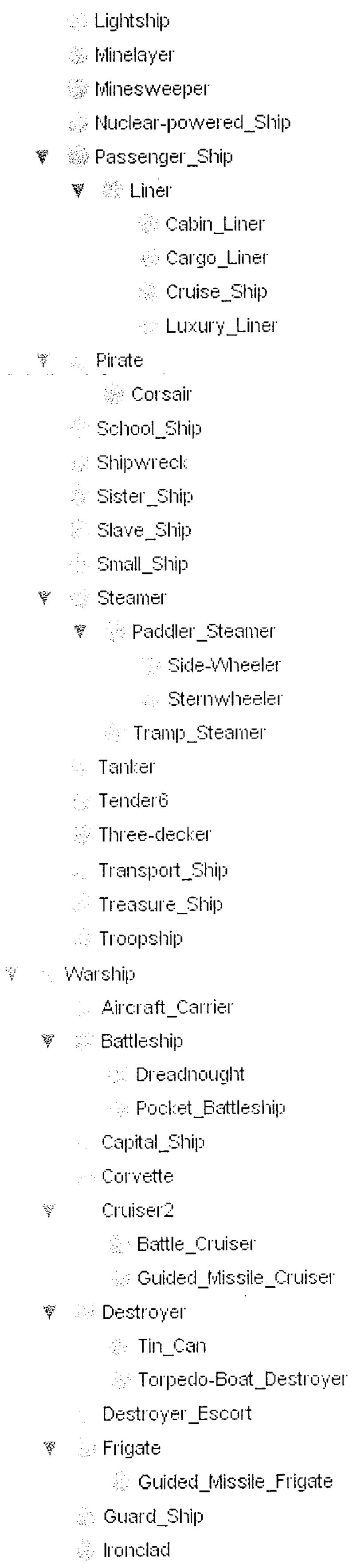


```
Outer_Space_Vehicle
    Rocket1
        Missile1
            Air-To-Air_Missile
            Air-To-Ground_Missile
            Ballistic_Missile
                Intercontinental_Ballistic_Missile
                    Minuteman
            Guided_Missile
                Antiballistic_Missile
                Buzz_Bomb
                Exocet
                Space_Probe
                Surface_To_Air_Missile
                    MANPAD
            Heat-Seeking_Missile
                Brilliant_Pebble
                Stinger
            Sidewinder
        Multistage_Rocket
        Test_Rocket
            Sounding_Rocket
    Spacecraft
        Lander
        Lunar_Excursion
        Space_Capsule
        Space_Shuttle
        Starship
        Vessel
            Bareboat
            Boat
                Ark
                Barge
                    Dredger
                    Houseboat
                    Pontoon
                    Scow2
                    Wherry1
                Bumboat
                Canal_Boat
                Ferry
                    Car-Ferry
                Fireboat
                Gondola
                Guard_Boat
                Gun_Boat
                Junk
```

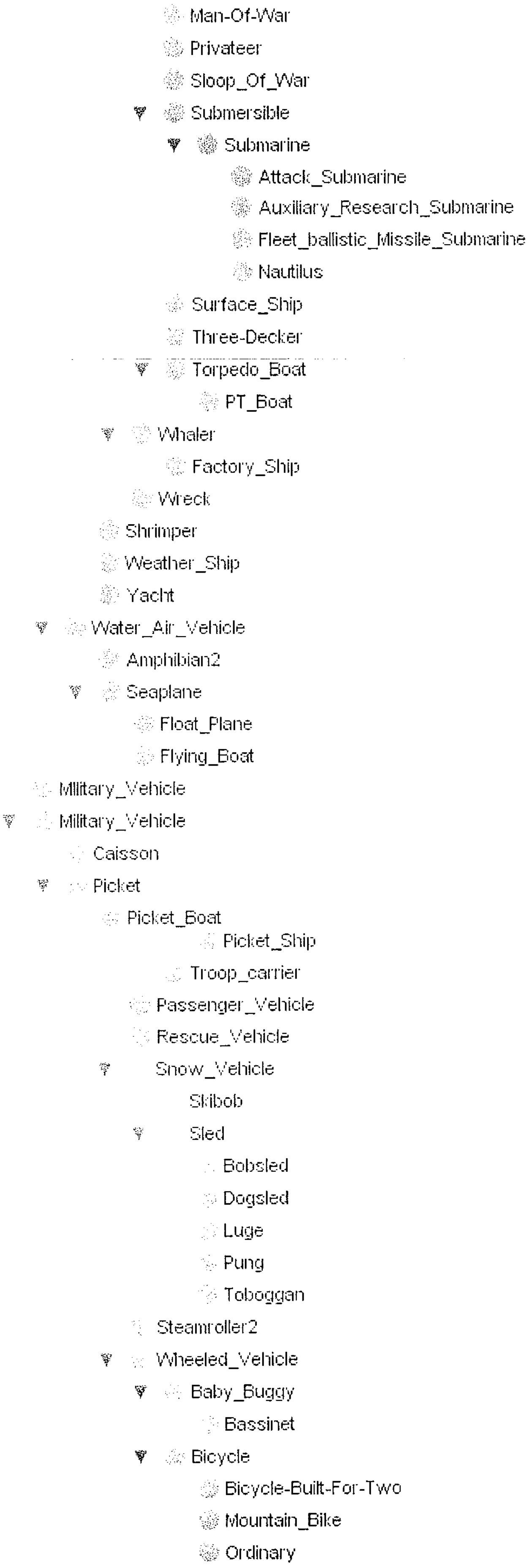


- Longboat
- Lugger
- Mackinaw
- Mailboat
- Motorboat
  - Cabin_Cruiser
  - Launch
  - Outboard_Motorboat
  - Speedboat
    - Hydrofoil
  - Water_Scooter
- Pilot_Boat
- Police_Boat
- Punt
- River_Boat
  - Keelboat
  - Showboat
- Scow1
- Sea_Boat
  - Lifeboat
  - Whaleboat
- Sharpie
- Small_Boat
  - Canoe
    - Birchbark_Canoe
    - Dugout
    - Kayak
    - Outrigger_Canoe
  - Cockleshell
  - Coracle
  - Dinghy
    - Rowing_Boat
    - Wherry2
  - Gig1
  - Racing_Boat
    - Racing_Gig
    - Shell
      - Racing_Skiff
      - Scull
  - Skiff
    - Sampan
  - Yawl1
    - Jelly_Boat
- Steamboat
- Surfboat
- Tender5
  - Gig4
- Tug_Boat

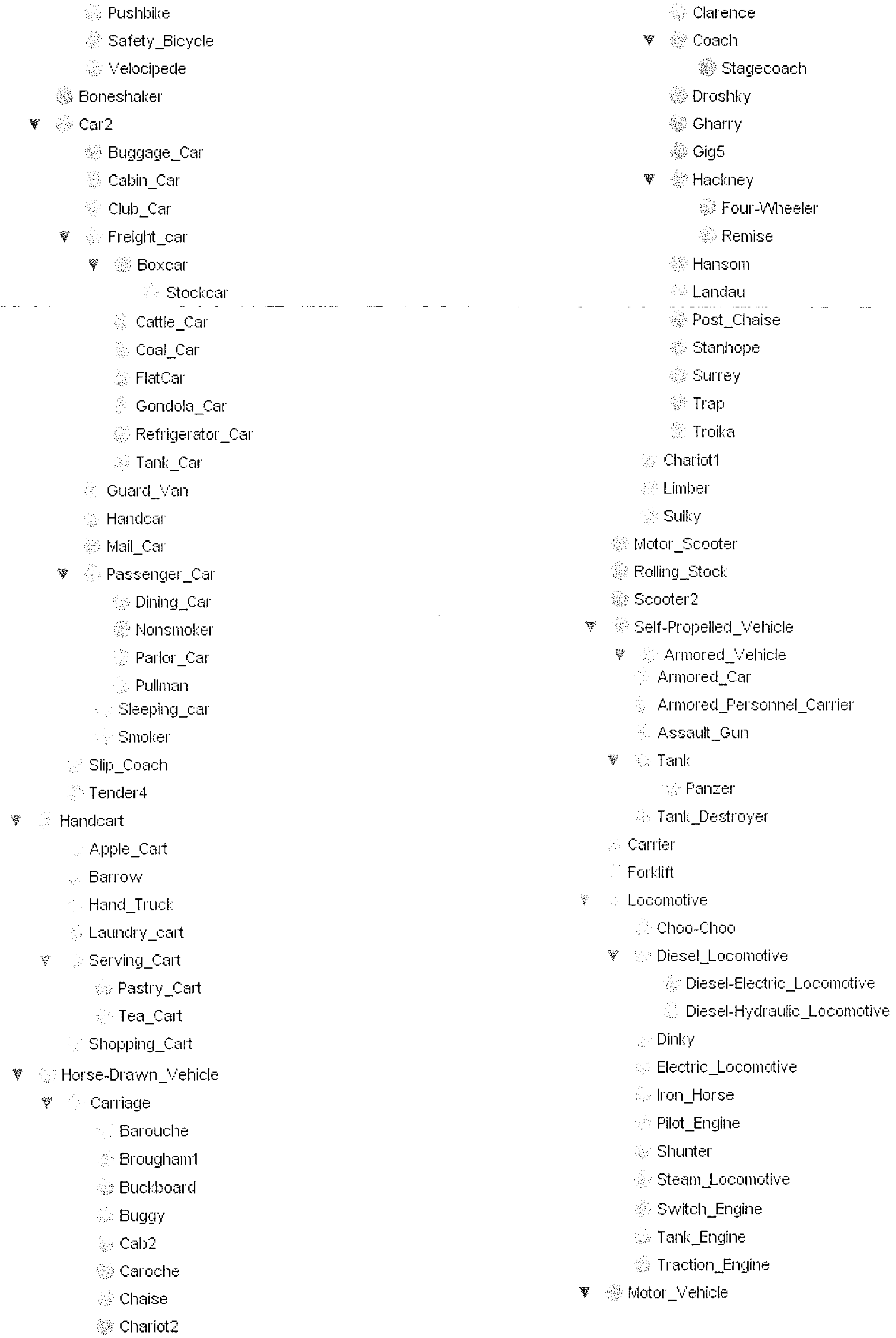
- Fishing_Boat
  - Trawler
- Galley
  - Trireme
- Iceboat
- Patrol_Boat
- Sailing_Vessel
  - Bark
  - Brig
  - Brigantine
  - Clipper
  - Cutter
  - Dhow
  - Felucca
  - Fore-And-After
  - Galleon
    - Carrack
  - Indiaman
  - Ketch
  - Rigger
  - Sailboat
    - Catamaran
    - Catboat
    - Trimaran
  - Schooner
    - Sharpshooter
  - Sloop
    - Knockabout
    - Raceabout
  - Smack
  - Square-rigger
  - Windjammer
  - Yawl2
- Ship
  - Abandoned_Ship
  - Blockade-Runner
  - Cargo_Ship
    - Banana_Boat
    - Bottom
    - Cattleship
    - Container_Ship
    - Liberty_Ship
    - Oil_Tanker
      - Supertanker
  - Flagship
  - Gas-Turbine_Ship
  - Hospital_Ship
  - Hulk
  - Icebreaker



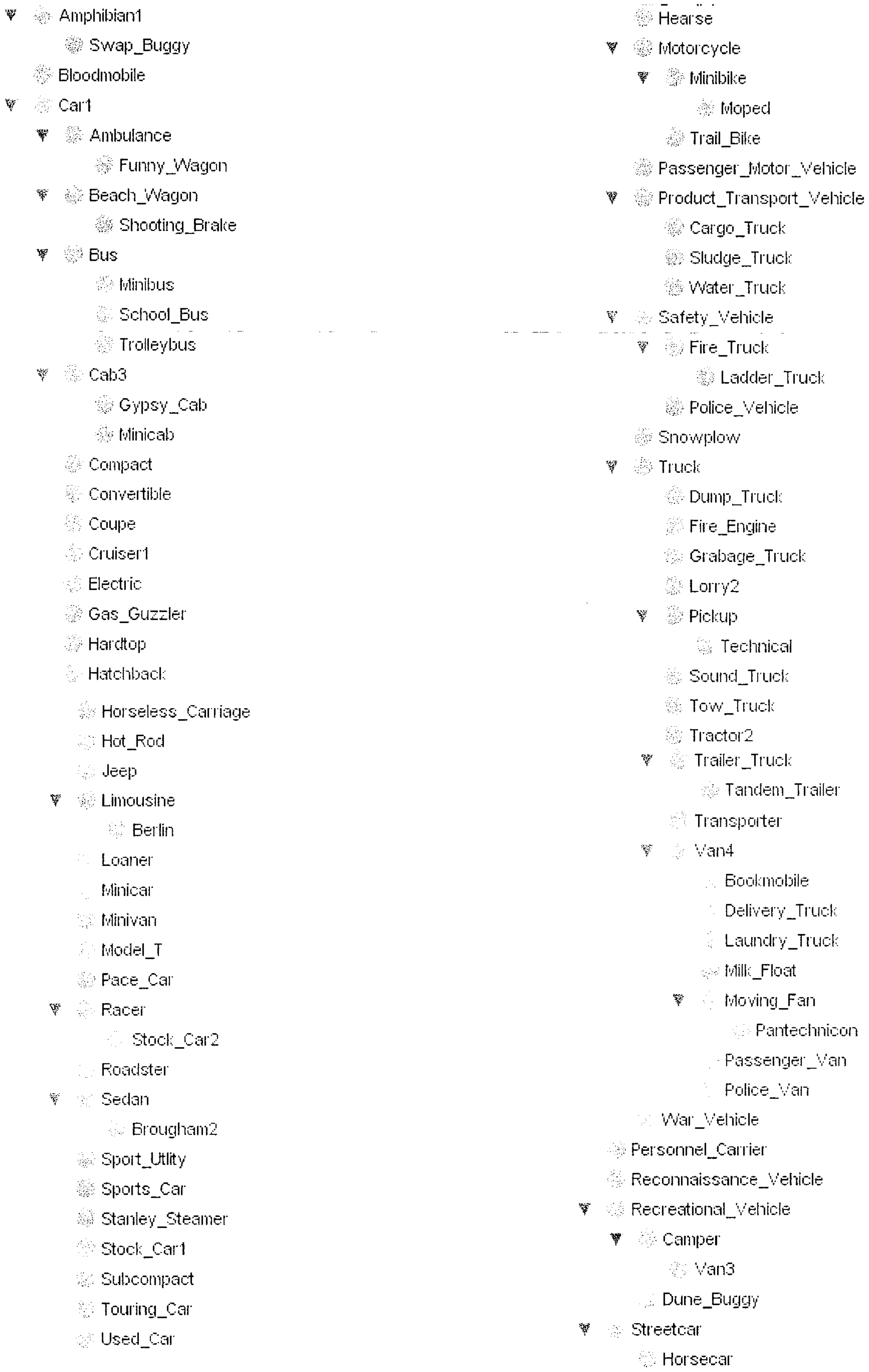
```
  Lightship
  Minelayer
  Minesweeper
  Nuclear-powered_Ship
  Passenger_Ship
    Liner
      Cabin_Liner
      Cargo_Liner
      Cruise_Ship
      Luxury_Liner
  Pirate
    Corsair
  School_Ship
  Shipwreck
  Sister_Ship
  Slave_Ship
  Small_Ship
  Steamer
    Paddler_Steamer
      Side-Wheeler
      Sternwheeler
    Tramp_Steamer
  Tanker
  Tender6
  Three-decker
  Transport_Ship
  Treasure_Ship
  Troopship
Warship
  Aircraft_Carrier
  Battleship
    Dreadnought
    Pocket_Battleship
  Capital_Ship
  Corvette
  Cruiser2
    Battle_Cruiser
    Guided_Missile_Cruiser
  Destroyer
    Tin_Can
    Torpedo-Boat_Destroyer
  Destroyer_Escort
  Frigate
    Guided_Missile_Frigate
  Guard_Ship
  Ironclad
```



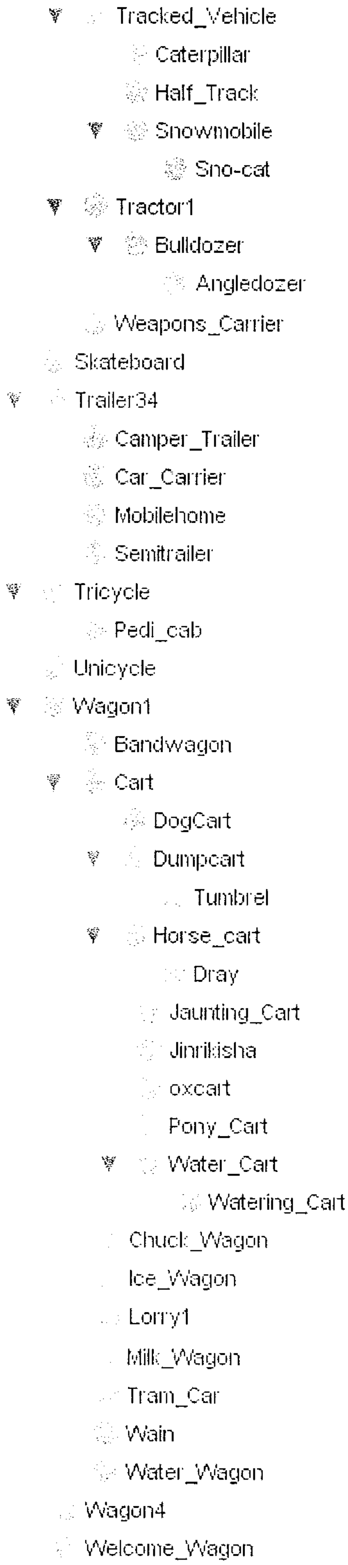
```
        Man-Of-War
        Privateer
        Sloop_Of_War
        Submersible
          Submarine
            Attack_Submarine
            Auxiliary_Research_Submarine
            Fleet_ballistic_Missile_Submarine
            Nautilus
        Surface_Ship
        Three-Decker
        Torpedo_Boat
          PT_Boat
      Whaler
        Factory_Ship
      Wreck
    Shrimper
    Weather_Ship
    Yacht
  Water_Air_Vehicle
    Amphibian2
    Seaplane
      Float_Plane
      Flying_Boat
Military_Vehicle
Military_Vehicle
  Caisson
  Picket
    Picket_Boat
          Picket_Ship
        Troop_carrier
      Passenger_Vehicle
      Rescue_Vehicle
      Snow_Vehicle
        Skibob
        Sled
          Bobsled
          Dogsled
          Luge
          Pung
          Toboggan
      Steamroller2
      Wheeled_Vehicle
        Baby_Buggy
          Bassinet
        Bicycle
          Bicycle-Built-For-Two
          Mountain_Bike
          Ordinary
```



  - Pushbike
  - Safety_Bicycle
  - Velocipede
- Boneshaker
- ▼ Car2
  - Buggage_Car
  - Cabin_Car
  - Club_Car
  - ▼ Freight_car
    - ▼ Boxcar
      - Stockcar
    - Cattle_Car
    - Coal_Car
    - FlatCar
    - Gondola_Car
    - Refrigerator_Car
    - Tank_Car
  - Guard_Van
  - Handcar
  - Mail_Car
  - ▼ Passenger_Car
    - Dining_Car
    - Nonsmoker
    - Parlor_Car
    - Pullman
  - Sleeping_car
  - Smoker
  - Slip_Coach
  - Tender4
- ▼ Handcart
  - Apple_Cart
  - Barrow
  - Hand_Truck
  - Laundry_cart
  - ▼ Serving_Cart
    - Pastry_Cart
    - Tea_Cart
  - Shopping_Cart
- ▼ Horse-Drawn_Vehicle
  - ▼ Carriage
    - Barouche
    - Brougham1
    - Buckboard
    - Buggy
    - Cab2
    - Caroche
    - Chaise
    - Chariot2
    - Clarence
    - ▼ Coach
      - Stagecoach
    - Droshky
    - Gharry
    - Gig5
    - ▼ Hackney
      - Four-Wheeler
      - Remise
    - Hansom
    - Landau
    - Post_Chaise
    - Stanhope
    - Surrey
    - Trap
    - Troika
  - Chariot1
  - Limber
  - Sulky
- Motor_Scooter
- Rolling_Stock
- Scooter2
- ▼ Self-Propelled_Vehicle
  - ▼ Armored_Vehicle
    - Armored_Car
    - Armored_Personnel_Carrier
    - Assault_Gun
    - ▼ Tank
      - Panzer
    - Tank_Destroyer
  - Carrier
  - Forklift
  - ▼ Locomotive
    - Choo-Choo
    - ▼ Diesel_Locomotive
      - Diesel-Electric_Locomotive
      - Diesel-Hydraulic_Locomotive
    - Dinky
    - Electric_Locomotive
    - Iron_Horse
    - Pilot_Engine
    - Shunter
    - Steam_Locomotive
    - Switch_Engine
    - Tank_Engine
    - Traction_Engine
  - ▼ Motor_Vehicle

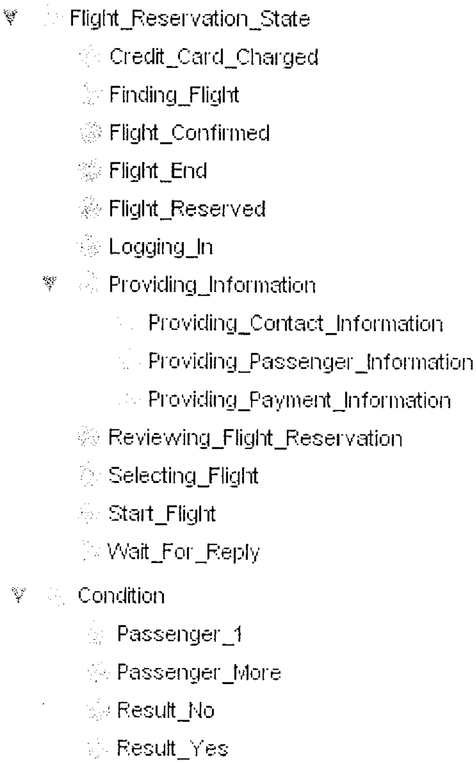


```
  Amphibian1
    Swap_Buggy
  Bloodmobile
  Car1
    Ambulance
      Funny_Wagon
    Beach_Wagon
      Shooting_Brake
    Bus
      Minibus
      School_Bus
      Trolleybus
    Cab3
      Gypsy_Cab
      Minicab
    Compact
    Convertible
    Coupe
    Cruiser1
    Electric
    Gas_Guzzler
    Hardtop
    Hatchback
    Horseless_Carriage
    Hot_Rod
    Jeep
    Limousine
      Berlin
    Loaner
    Minicar
    Minivan
    Model_T
    Pace_Car
    Racer
      Stock_Car2
    Roadster
    Sedan
      Brougham2
    Sport_Utlity
    Sports_Car
    Stanley_Steamer
    Stock_Car1
    Subcompact
    Touring_Car
    Used_Car
  Hearse
  Motorcycle
    Minibike
      Moped
    Trail_Bike
  Passenger_Motor_Vehicle
  Product_Transport_Vehicle
    Cargo_Truck
    Sludge_Truck
    Water_Truck
  Safety_Vehicle
    Fire_Truck
      Ladder_Truck
    Police_Vehicle
  Snowplow
  Truck
    Dump_Truck
    Fire_Engine
    Grabage_Truck
    Lorry2
    Pickup
      Technical
    Sound_Truck
    Tow_Truck
    Tractor2
    Trailer_Truck
      Tandem_Trailer
    Transporter
    Van4
      Bookmobile
      Delivery_Truck
      Laundry_Truck
      Milk_Float
      Moving_Van
        Pantechnicon
      Passenger_Van
      Police_Van
  War_Vehicle
Personnel_Carrier
Reconnaissance_Vehicle
Recreational_Vehicle
  Camper
    Van3
  Dune_Buggy
Streetcar
  Horsecar
```



```
    Tracked_Vehicle
        Caterpillar
        Half_Track
        Snowmobile
            Sno-cat
    Tractor1
        Bulldozer
            Angledozer
    Weapons_Carrier
Skateboard
Trailer34
    Camper_Trailer
    Car_Carrier
    Mobilehome
    Semitrailer
Tricycle
    Pedi_cab
Unicycle
Wagon1
    Bandwagon
    Cart
        DogCart
        Dumpcart
            Tumbrel
        Horse_cart
            Dray
        Jaunting_Cart
        Jinrikisha
        oxcart
        Pony_Cart
        Water_Cart
            Watering_Cart
    Chuck_Wagon
    Ice_Wagon
    Lorry1
    Milk_Wagon
    Tram_Car
    Wain
    Water_Wagon
Wagon4
Welcome_Wagon
```

Figure A4 Vehicle ontology



```
Flight_Reservation_State
    Credit_Card_Charged
    Finding_Flight
    Flight_Confirmed
    Flight_End
    Flight_Reserved
    Logging_In
    Providing_Information
        Providing_Contact_Information
        Providing_Passenger_Information
        Providing_Payment_Information
    Reviewing_Flight_Reservation
    Selecting_Flight
    Start_Flight
    Wait_For_Reply
Condition
    Passenger_1
    Passenger_More
    Result_No
    Result_Yes
```

Figure A5 Travel state ontology

Conceptual Modeling and Reasoning using Ontology

Ong Siew Kin
Computer Aided Translation Unit
Universiti Sains Malaysia, 11800 Minden, Penan, Malaysia
skin@cs.usm.my

Tang Enya Kong
Computer Aided Translation Unit
Universiti Sains Malaysia, 11800 Minden, Penan, Malaysia
enyakong@cs.usm.my

ABSTRACT

Electronic marketplace consists of many competing companies who supply each other with web resources. The heterogeneity of resources has caused structural and semantic integration problems between resource provider and requester. To solve these problems, the Semantic Web community has initiated the Semantic Web research and development where ontology has been proposed to model the Web resources. In the electronic marketplace, ontology can act as a multi-level adaptability base to serve different providers, end users, devices and contexts. In this paper, we show the usefulness of ontology in concept modeling of Web resources and how reasoner like Algernon can be used to derive additional information from rules augmented in the ontology.

KEYWORDS

Knowledge Representation, Ontology, Semantic Web, Reasoning, Conceptual Modeling

1. Introduction

Electronic marketplace consists of many competing companies who supply each other with web resources. The Web resources (products or services) are widely distributed, highly heterogeneous, diverse and autonomous. The rapid growth of the number of Web resources makes finding, selecting and combining them a desirable but challenging task. During the business negotiation phase, both the resource provider and requester publish their own product (resource) catalogs on the marketplace. It is most likely that the catalogs are in different formats or terms and do not match. Therefore, intermediate Web catalogs are needed to provide matchmaking. This requires standardized ontologies to capture the resource descriptions (See figure 1). We are proposing the aggregator module as a virtual workbench or middleware framework to integrate and capture the semantics of the resources. The framework will take an ontological approach in representation formalism. It provides an ontological approach in structuring the product catalog. Due to the ontology powerful knowledge formalism and associated inference mechanisms, ontology-based representation will be use as the base for query formation as well as matchmaking. In this paper, we focus on conceptual modeling and rule reasoning. We have designed a small ontology in our prototype system for illustration.

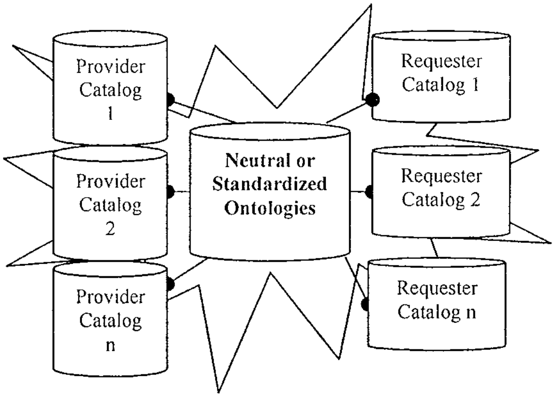


Figure 1. Heterogeneous Resource Integration

The rest of the paper is structured as follows. Section 2 and Section 3 will give background information on ontology and Web ontology languages. In section 4, we will brief on the available ontology tools. We show how reasoning and rules are used for deriving additional information in Section 5 before we conclude the paper in Section 6.

2. Ontology

The widely accepted definition of ontology is the one given by Gruber [1]: "an explicit specification of conceptualization". Gruber uses the description given in his paper [2] to further refine the term conceptualization as "the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them". One of the primary purposes of constructing ontology is to provide a standard, unambiguous representation of a particular domain of knowledge. It is not a new issue since there are ontologies being used in the computing environment nowadays, for example the taxonomies categorizing web sites (Yahoo! Directory), categorizations of products for sale and their features (Amazon.com), large, standardized vocabularies such as SNOMED, ontology of terminologies for products and services (UNSPSC).

Different categorization structures exist in representing web resources, ranging from the simplest flat set (no category), strict hierarchical tree (single classification of products), directed acyclic graph (multiple classification of products) to the more expressive taxonomy (spanning tree, multiple inheritance, bi-directional cross-references, synonymous and translations). Simple category is easy to set up, however simple category lack the rigorous logics for machines to make inference from. Most of the early ontologies do not provide an explicit hierarchy except capturing many of the "naturally occurring ontologies"[3]. This means the hierarchy is not a strict subclass or "isa" hierarchy. In our work, we follow strict hierarchical subclass relationships between classes to be beneficial from the subsumption reasoning and inheritance mechanism. Besides, ontology should have:
Finite controlled (extensible) vocabulary set
Unambiguous interpretation of classes and term relationships. Each interpretation is unique with the use of namespace (uniform resource locator)

From the definition, ontology defines the terms and concepts used to describe an area of knowledge, as well as the relationships among them. It includes concepts, relationship between concepts, properties, property values, constraints and rules of those concepts. The shared vocabulary model of ontology is less expressive than free-text description. However, it gives better search and enables reasoning. Designing a stable ontology requires time and efforts, but once the stable ontologies have been developed, the ontologies will be the universal baseline for information systems. By using ontology as the universal representation, both providers and requesters are independent from the terms used to describe their resources. The hierarchical representation of ontology also gives better navigation mechanism in query specialization and generalization. This progressive browsing mechanism allows the user to either drilling down or rolling up along the concept hierarchy that best suits its purposes and situations.

3. Web Ontology Language

The Web language in the electronic world is XML. It gives a standardized way to describe tree structures with a linear syntax. However, it only tackles the syntactic interoperability problem. A Document Type Declaration (DTD) is used to add constraints to the tags and structure in an XML document. The use of XML together with DTD does help standardize the Web resources, but they are limited in their expressive power and so they have been extended by a more advanced W3C standard, XML Schemas and later the Resource Description Framework (RDF) and Resource Description Framework Schema (RDFS). Still, the expressiveness of the languages is still limited. This gives rise to the more advanced web languages such as DAML+OIL and the most recent OWL.

Our ontology is written in the Web Ontology Language (OWL), specifically OWL DL, a semantic markup language developed as a vocabulary extension of the previous ontology web language RDF and DAML+OIL. OWL is part of the growing stack of W3C recommendations related to Semantic Web proposed by W3C Community [4]. An OWL document consists of optional ontology headers plus any number of class axioms, property axioms, and facts about individuals (instances). Class is the central entity that provides an abstraction mechanism for grouping resources with similar characteristics. Every OWL class is associated with a set of individuals, called the class extension, class instances or class members. There are six ways to define class descriptions:
A class identifier
An exhaustive enumeration of individuals that form the instances of a class
A property restriction
The intersection of two or more class descriptions
The union of two or more class descriptions
The complement of a class description

```
<owl:Class rdf:ID="Resource"/>
```

```
<owl:Class rdf:ID="Car">
  <rdfs:subClassOf rdf:resource="#Resource"/>
</owl:Class>
<owl:Class rdf:ID="Toyota">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Car"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Honda">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Car"/>
  </rdfs:subClassOf>
</owl:Class>
<owl:Class rdf:ID="Camry">
  <rdfs:subClassOf rdf:resource="#Toyota"/>
</owl:Class>
  <owl:Class rdf:ID="Unser">
    <rdfs:subClassOf>
      <owl:Class rdf:about="#Toyota"/>
    </rdfs:subClassOf>
  </owl:Class>
<owl:Class rdf:ID="Civic">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Honda"/>
  </rdfs:subClassOf>
<owl:Class rdf:ID="Altis">
  <rdfs:subClassOf>
    <owl:Class rdf:about="#Toyota"/>
  </rdfs:subClassOf>
</owl:Class>
```

Figure 2. OWL Code Of Simple Car Concept

Figure 2 shows the fragment of OWL code to represent Car concept. From the illustration, Car is a kind of Resource, Toyota and Honda are some types of Car. Altis, Camry, and Unser are some kinds of Toyota. OWL code can be generated using any text editors; however tools exist to make the work easier especially graphical tool. In section 4, we use Protégé 2000 as tool for coding and viewing OWL file.

4. Ontology Tools

There are many toolkits for ontology development and deployment. Protégé-2000 [5], OILed [6] and OntoEdit [7] are tools for ontology creation and knowledge acquisition. They support ontologies editing, check for inconsistencies using reasoner and provide mappings among different ontologies. Ont-O-Mat [8], on the other hand, focuses on the issue of integration with the Web. SMORE, the Semantic Markup, Ontology, and RDF Editor provides users an integrated environment for creating Web pages, email, and other online content while making use of ontology semantic.

To construct and maintain the ontology, a flexible and extensible tool is needed. Protégé 2000, developed by the Stanford University School of Medicine, provides the right environment for developing and maintaining the ontology in various ontology languages. Its graphical user interface (GUI) provides a better interface to deal with the language syntax. In our work in developing OWL ontology, we need to have protégé 2000 with the OWL plug-in. OWL plug-in is a comprehensive OWL editor based on the Protégé ontology development platform. It can be used to load and save OWL files in various formats, to edit OWL ontologies with custom-tailored graphical widgets, and to perform intelligent reasoning based on Description Logics (DL). The reasoning engine that is provided directly by the plug-in is RACER [9]. With the help of the reasoner, we can infer implicit knowledge from the given ontology. The current user interface supports two types of DL reasoning: consistency checking and classification (subsumption).

5. Reasoning And Rules

The power of OWL is the language expressive representative formalism and reasoning power. Because OWL was derived from DAML+OIL, it can take advantage of the existing reasoning algorithms in Description Logics (DL). There are four basic tasks of terminology reasoning in DL [10]:

- Subsumption check – check if one class is a subclass of another
- Consistency check – check for inconsistent class definitions
- Taxonomy construction – computes all subclass relations (deriving new relations from the necessary and sufficient conditions defined for a class)
- Classification – determines the classes that immediately subsume or are subsumed by a given class.

Reasoning tasks involving individuals, which are called assertion reasoning are:

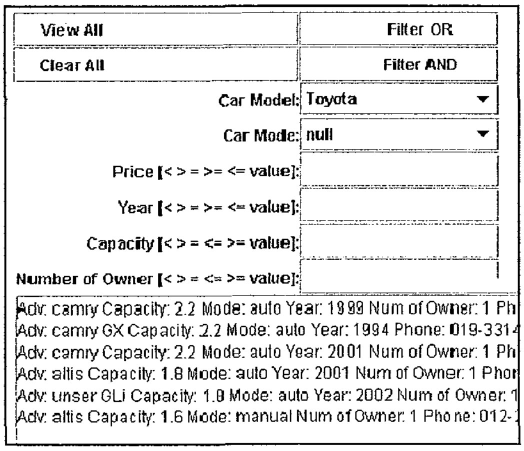
- Realization – given a partial description of an individual, find the most specific concept that describes it.
- Instance checking – given a partial description of an individual and a class description, finds whether the class describes the instance
- Individual retrieval – finds all individuals that are described by a given concept.

Reasoning tasks are important during ontology development and deployment. It checks for concept inconsistencies, hidden dependencies, redundancies, misclassification and derive implicit relationships during development of ontologies. While deployment, it is useful to answer queries as well as doing classifying. The semantic of OWL allows us to define a ranking function that distinguishes multiple degrees of matching. There are three types of matches: exact match where the concept to be found is found, plug-in match where the concept to be found is more specific than the concept in ontology, and subsume match where the concept to be found is more general than the concept in ontology. The scoring function of matching degree is given below:

Exact Match > Plugin Match > Subsume Match

The matching degree could be used is query answering as used in our ontology. Some examples are:

Query: Search for advertisement on all Toyota model. The search will return all direct and indirect instances of Toyota (including the instances of Camry, Altis and Unser).



Car Model: Toyota

Car Mode: null

Price [< > = >= <= value]:

Year [< > = >= <= value]:

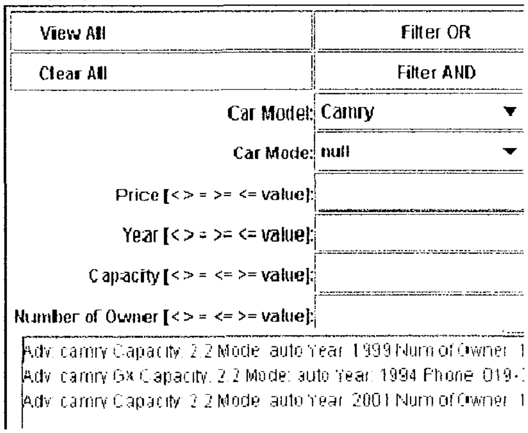
Capacity [< > = <= >= value]:

Number of Owner [< > = <= >= value]:

```
Adv: camry Capacity: 2.2 Mode: auto Year: 1999 Num of Owner: 1 Ph
Adv: camry GX Capacity: 2.2 Mode: auto Year: 1994 Phone: 019-3314
Adv: camry Capacity: 2.2 Mode: auto Year: 2001 Num of Owner: 1 Ph
Adv: altis Capacity: 1.8 Mode: auto Year: 2001 Num of Owner: 1 Phor
Adv: unser GLi Capacity: 1.8 Mode: auto Year: 2002 Num of Owner: 1
Adv: altis Capacity: 1.6 Mode: manual Num of Owner: 1 Phone: 012-
```

Figure 3. Algernon Rule Triggering

Query for advertisement on Toyota Camry vehicle ontology. The search agent has been written as a java plug-in tab in Protégé environment.



Car Model: Camry

Car Mode: null

Price [< > = >= <= value]:

Year [< > = >= <= value]:

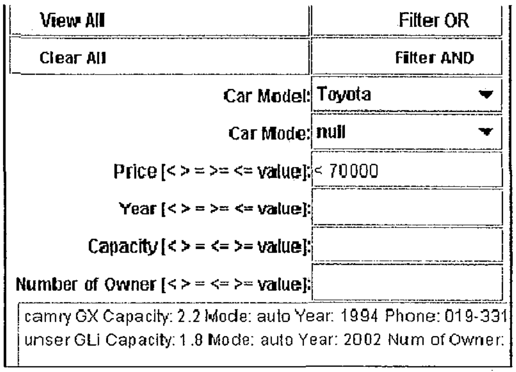
Capacity [< > = <= >= value]:

Number of Owner [< > = <= >= value]:

```
Adv: camry Capacity: 2.2 Mode: auto Year: 1999 Num of Owner: 1
Adv: camry GX Capacity: 2.2 Mode: auto Year: 1994 Phone: 019-
Adv: camry Capacity: 2.2 Mode: auto Year: 2001 Num of Owner: 1
```

Figure 4. Algernon Rule Triggering

Search for advertisement on all Toyota models that is less than RM 70000.



Car Model: Toyota

Car Mode: null

Price [< > = >= <= value]: < 70000

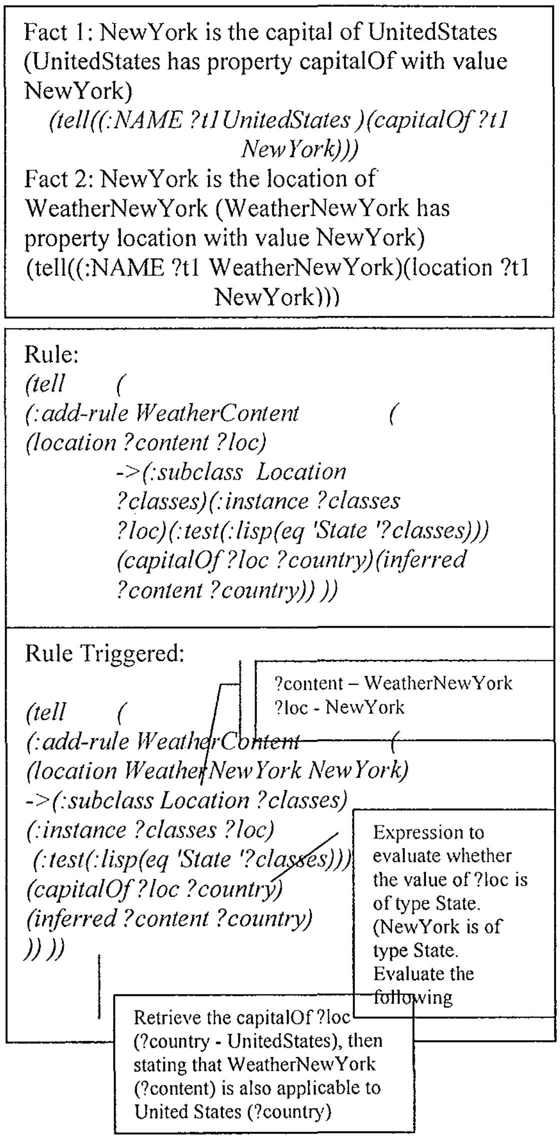
Year [< > = >= <= value]:

Capacity [< > = <= >= value]:

Number of Owner [< > = <= >= value]:

```
camry GX Capacity: 2.2 Mode: auto Year: 1994 Phone: 019-331
unser GLi Capacity: 1.8 Mode: auto Year: 2002 Num of Owner:
```

Figure 5. Algernon Rule Triggering



Fact 1: NewYork is the capital of UnitedStates (UnitedStates has property capitalOf with value NewYork)

(tell((:NAME ?t1 UnitedStates)(capitalOf ?t1 NewYork)))

Fact 2: NewYork is the location of WeatherNewYork (WeatherNewYork has property location with value NewYork)

(tell((:NAME ?t1 WeatherNewYork)(location ?t1 NewYork)))

Rule:

(tell (
(:add-rule WeatherContent (
(location ?content ?loc)
->(:subclass Location ?classes)(:instance ?classes ?loc)(:test(:lisp(eq 'State '?classes)))
(capitalOf ?loc ?country)(inferred ?content ?country))))

Rule Triggered:

(tell (
(:add-rule WeatherContent (
(location WeatherNewYork NewYork)
->(:subclass Location ?classes)
(:instance ?classes ?loc)
(:test(:lisp(eq 'State '?classes)))
(capitalOf ?loc ?country)
(inferred ?content ?country)
))))

?content – WeatherNewYork
?loc - NewYork

Expression to evaluate whether the value of ?loc is of type State. (NewYork is of type State. Evaluate the following

Retrieve the capitalOf ?loc (?country - UnitedStates), then stating that WeatherNewYork (?content) is also applicable to United States (?country)

Figure 6. Algernon Rule Triggering

In addition to the concepts and properties defined in the ontology, one can define rules to express knowledge [11]. The simplest variant of such rules are expressions of the form C → D, where C and D are distinct concepts. The meaning of such a rule is "if an individual is proved to be instance of C, then derive that it is also an instance of D". Such rules are often called trigger rules. Access-limited logic and its embodiment in a knowledge-representation language named Algernon [12] is a language for representing knowledge in the computer, and a method for drawing conclusions and answering questions from that knowledge. We have used Algernon plug-in in Protégé2000 to derive additional information. By adding rules into ontology, new information can be derived. Algernon's syntax is like a predicate. It has the form of (slot frame value). To express F is A's mother's brother's cousin's wife's sister. We write the Algernon facts:
A mother B brother C cousin D wife E sister F

For example in our prototype, one provider may supply weather information of New York using the terms from the ontology (UnitedStates, NewYork, WeatherNewYork are instances while capitalOf, inferred and location are object properties). By running the Algernon engine, the weather of United States could be inferred based on the relationship stating that New York is the capital city of United States. See figure 6.

6. Conclusion

To achieve more intelligent information system, ontology is needed to define terminology that could be used for the metadata description. One the metadata is captured, instances of concept are populated. After the instances of concepts have been created, the instances could be exported to the Web server in various presentation views to be found by the search agents. Future works will focus on ontology construction and instance population.

7. Acknowledgements

This work has been funded by iNavigate Sdn Bhd under Multimedia Super Corridor Research & Development Grant Scheme (MGS), as part of the m-Financial™ project. The idea and concept of this research work was originated by KK Lim, one of the iNavigate's founder. m-Financial™ middleware is part of the m-Apps(tm) middleware technology being developed by iNavigate. This paper was written after several technical discussions held with Joseph Yeong, another founder of iNavigate.

References

1. T. R. Gruber, "A Translation Approach to Portable Ontologies", Knowledge Acquisition. 5(2): pp. 199-220. 1993
2. Genesereth M.R. and N.J. Nilsson, "Logical Foundation of Artificial Intelligence". Los Altos, California. 1987
3. Ora Lassila and Deborah L. McGuinness,, "The Role of Frame-Based Representation on the Semantic Web". Electronic Transactions on Artificial Intelligence 2001. Volume 5 Number: 2001-03-07, 1403-2031 (Printed version) 1403-204X (Electronic version). 2001
4. Berners-Lee, T. and M. Fischetti, 1999. "Weaving thc Web: The Original Design and Ultimate Destiny of the World Wide Web by its Inventor". Harper, San Francisco. 1999.
5. M.A.Musen, R.W.Fergerson, W.E. grosso, N.F.Noy, M.Crubezy, & J.H.Gennari, "Component-Based Support for Building Knowledge-Acquisition Systems". Conference on Intelligent Information Processing (IIP 2000) of the International Federation for Information Processing World Computer Congress (WCC 2000). Beijing, China. 2000.
6. Sean Bechhofer, Ian Horrocks, Carole Goble, Robert Stevens. "OilEd: a Reason-able Ontology Editor for the Semantic Web". Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September 19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396--408. 2001
7. Sure, Y Et Al, "OntoEdit: Collaborative Ontology Development for the Semantic Web". Proceedings of the first International Semantic Conference 2002 (ISWC 2002). Sardinia, Italia. 2002
8. Handschuh, S. Staab, S.Madche, "CREAM – Creating relational metadata with a component-based, ontology driven annotation framework". K-CAP. 2001
9. Volker Haarslev and Ralf Moeller "RACER user's guider and reference manual", 2003, http://www.cs.concordia.ca/~faculty/haarslev/racer.
10. Javier Gonzalez-Castillo, David Trastour, Claudio Bartoline, "Description Logics for Matchmaking of Services". HP Lab Bristol. 2001
11. Franz Baader, Alex Borgida, Ralph Kuesters and Deborah L. McGuinness, "Matching in Description Logics". In Journal of Logic and Computation -- Special Issue on Description Logics. Volume 9, number 3. 1999
12. A B. J. Kuipers and J. M. Crawford. 1994. "Short Algernon Reference Manual

Creating and Organising Semantic Web Services

Ong Siew Kin and Tang Enya Kong
UTMK, School of Computer Sciences,
Universiti Sains Malaysia
{skin, enyakong}@cs.usm.my

Abstract

The vision of Semantic Web Services (SWS) is to create dynamically interoperating nodes on the web. To help realising the SWS, our work proposes a model to organise the semantic web services. Besides, we propose a method to transform existing web sites into semantic web services. We achieve this by defining our domain and state ontologies, analysing the navigational behaviors of web sites using state-charts and instantiating semantic web services by annotating them with state ontologies and domain ontologies.

Keywords
Semantic Web, Ontology, Semantic Web Services

1. INTRODUCTION

The vision of SWS is to create dynamically interoperating nodes on the web. In supporting of this effort, technologies from Web Services and Semantic Web have been tailored towards this vision. These include XML standard as the unifying factors, SOAP as the message passing definition, WSDL as the description language for service interface, and BPEL4WS as the mechanisms for describing interaction among several web services. On the other hands, the Semantic Web offers the ontological concepts in annotating web service capabilities as well as interactions. Therefore, SWS can be seen as using ontologies and semantically annotated service descriptions to automate the fulfillment of tasks and transactions [9]. Our work is in the same direction as the proposed SWS, we propose a model to organize the semantic web services and we aim to fasten the adoption of SWS by transforming current web sites (offering certain services through HTML forms) into semantically annotated web services. Section two of the paper outlines our proposed SWS organising model. Section three describes the requirements for web service annotation and section four briefs the creation of our domain ontologies. The process of wrapping web sites into semantic web services is outlined in section five before we give some related work and conclude our paper.

2. WEB SERVICES ORGANIZATION

Inspired by the work [2] on layered service ontology, we have taken some aspects of the layering to cater for two-dimensional view on web services. Figure 1 gives the proposed SWS model.

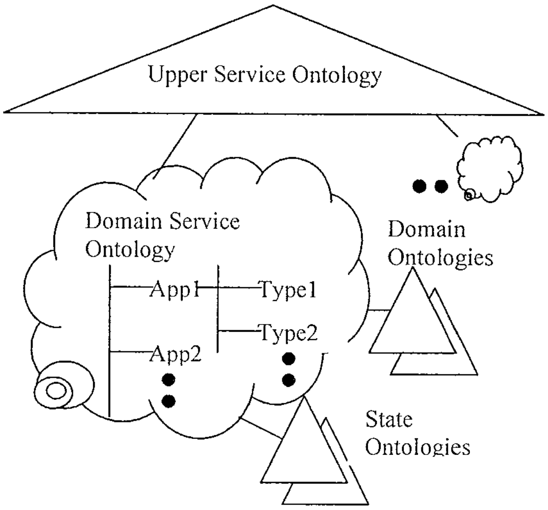


Figure1 SWS organization

Upper service ontology provides the basic structure for web service description. DAML-S [1] and its successor OWL-S [10] are potential candidates. These ontologies provide a set of basic markup primitives that could be used to describe different attributes of web services. Our model organises semantic web services by domain service ontology. Community from the same domain can usually reach consensus easily on matters like vocabularies (domain ontologies) used to communicate, security policies and communication protocols. In our work, we assume that different parties from the travel domain have formed a community and are initiating the semantic web services organisation.

There are two different views in analysing processes [12]: parts of a process and types of a process. In our approach, we view web services as processes and organise them by part and by type. Figure 2 shows the service category for the Travel domain.

Transportation Service and Accommodation Service are some "parts" (sub activities) of the travel service (In planning a travel, there are some sub activities involved in getting services from transportation and accommodation). Transportation Service is further sub-typed into Flight Service, Car Service and Train Service. Organising the services in this way facilitates the discovery (browsing and searching) and composition of semantic web services [11].

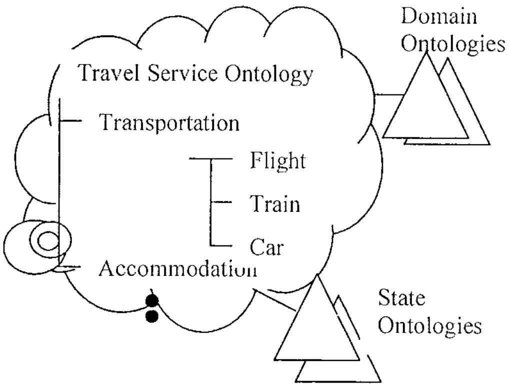


Figure 2 SWS model for travel domain

The third element in the model is made up of many domain specific ontologies and state ontologies. This group of ontologies provides common vocabularies for describing the services.

3. WEB SERVICES DESCRIPTION

There are many ways to describe a web service. The quality of service description will affect directly its applicability. By just describing the service name and its provider, we can do a simple keyword-search based on these criteria. By adding semantic to the service description (marking with ontology terms), searching on the services can now be based on semantic and not just pure keywords. Annotating service with attributes like input, output, precondition and effect of the service will enable a functional search and simple composition. To allow the composing agent to automatically compose and further execute the service, the internal behaviors and the binding information of a service must be captured. Therefore, we realize that specification of semantic web services which aim at automatic discovery, composition, execution and monitoring should include semantic information about both data flow (functionalities) and control flow (behaviors). Besides describing the basic interface of web services as proposed by W3C in DAML-S, our work use state-charts [8] as the workflow specification language. We use DAML-S in this paper although OWL-S is the newest recommendation. In the future, we will use OWL-S instead.

4. DOMAIN ONTOLOGY CREATION

In Semantic Web, we need domain ontologies that organise the most relevant concepts, their relationships, and instances. In SWS, ontologies provide a shared conceptualization for interpreting semantic markup of web services. It enables services across the web to use the same terminologies to interpret each other's message and behavior description.

In our work, we need ontologies in domain relevant to travel as well as state ontologies that describe the various stages of transaction. In particular, we need to describe service input, output, pre-condition and effect with ontology vocabularies. The state ontology is created after analysing the web sites by collecting the states from the state-charts. The remaining text in this section refers to creation of domain ontologies.

Creating ontological descriptions for real-world information is nontrivial. There are some available public domain ontologies and communities in electronic business have created several standards [5] (product, catalog and document structure standard) in effort to promote information integration. An example of standards in identifying e-commerce products, United Nation Standard Products and Services Codes (UN/SPSC) defines a hierarchical classification with five levels with each level containing a two-character numerical value and a textual description.

While creating ontologies for use in our work, we reuse some terminologies from the UN/SPSC and enhance it further with WORDNET [3]. There are several issues to be considered for a successful ontology such as:

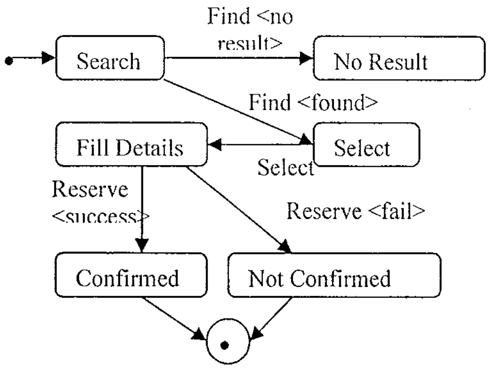
- Industry specific need have to be taken into account. Essential attributes and values of a product highly differ per industry.
- Country specific needs and regulations
- Should be broadly supported by standardisation bodies.
- Needs broad industry support (from both businesses, marketplaces, and e-commerce software vendors)

By considering these issues, our SWS model categorises semantic web services by domain community to ease the creation and utilisation of ontologies.

5. INSTANTIATING THE SEMANTIC WEB SERVICES

Before instantiating semantic web services, we need to acquire the generic service description from the current web sites. We do this by analysing the navigation behavior of web sites using state-charts. Various states identifying in the state-charts makes up the content for state ontology. Generalization technique [6] is applied on the collected state-charts to create a generic service behavior for similar services. These generic service behaviors will be the attributes of services as defines in our service category ontology.

After getting the generic service description, we populate the service model with real semantic web services using the wrapping technique. Each web site is categorised under a certain service category, which imports several states from the state ontology. Service instance is further marked up using terminologies from the domain ontologies to describe its required input and output. Each corresponding state-charts will be transformed into workflow language (BPEL4WS) and its pre-state and post-state will become its pre-condition and effect respectively. Figure three illustrates how a start-chart diagram can be transformed into a generic service description (input, output, pre-condition end effect). State-chart provides important information about service pre-condition and effect but not input and output information. Input and output information could be obtained by analysing web elements such as web form elements and returned result. Each event in the state-chart is the function in our service description (Notice that we add two events into every service description (init() and end() to denote the start and the end of service flow).



Application: Transportation
Type: Car Rental
Operation:

- Init (I, O, P, E: Search)
- Find (I, O, P: Init, E: Select OR No_Result)
- Select (I, O, P: Select, E: Fill_Details)
- Reserve (I, O, P: Fill_Details, E: Confirmed OR Not_Confirmed)
- End (I, O, P: Confirm OR Not_Confirmed, E)

I: Input; O: Output; P: Pre-condition; E: Effect

Figure 3 Creation of generic semantic web service from state-chart

6. RELATED WORK

There are many papers [3, 7] that describe the creation semantic web services; however they focus mainly on service profile and not on behavior attributes. Our work focuses on describing a web service via its input, output, pre-condition and effect (IOPE) besides the basic profile attributes. We contribute by proposing a way to wrap existing web applications into semantic web services, thus defining its IOPE especially in defining the pre-condition and effect of the services. This differs from the current work that does not relate to the existing web applications. Our approach creates the service description as discussed and fastens the process of realising the semantic web services.

Ontology creation can be done by human expert. However, there are research work aims at generating ontologies automatically or semi-automatically which based on table analysis [13], taxonomy-directed web sites [4] and thesaurus extension [2]. These efforts complement with our work since we aim to utilize ontologies that are commonly accepted. For state ontology, none of the above-mentioned work is creating ontology for describing states.

7. CONCLUSION

To realise the vision of SWS, there is a need for an organising model. We propose a model that facilitates the creation, discovery and composition of semantic web services. By focusing on domain classification, agreement on ontologies can be easily achieved and commitment on the ontologies is better secured. We also contribute in fastening the creation of semantic web services. By using state-charts as the analysis tool, we derive the generic service description that captures the behavior attributes of services. Our future work will further evaluate how our model can facilitate the composition of semantic web services.

REFERENCES

[1] Anupriya Ankolekar, Mark Burstein, Jerry Hobbs J., et al. "DAML-S: web service description for semantic web", First International Semantic Web Conference (ISWC02), 2002.

[2] B. J. Wielinga, A. Th. Schreiber, J. Wielemaker and J. A. C.Sandberg, K-CAP 2001: Proceedings of the international conference on Knowledge Capture, NY: ACM Press, 2001, Chapter From Thesaurus to Ontology, pp. 194-201.

[3] Christiane Fellbaum, Wordnet: an electronic lexical database (language, speech, and Communication, The MIT Press, 1998.

[4] Davalcu, H., Vadrevu, S., Nagarajan, S. and Ramakrishnan, I.V., IEEE Intelligent Systems and Their Applications , IEEE Intelligent System, 2003, Chapter OntoMiner: bootstrapping and populating ontologies from domain-specific Web sites, pp. 24-33.

[5] Fairchild, A.M., and de Vuyst, B., "Coding Standards Benefiting Product and Service Information in E-commerce", 35th Annual Hawaii International Conference on System Sciences (HICSS'02), 7-10th Jan 2002. vol. 8, pp. 3201-3208.

[6] George M.Wyner and Jiantae Lee, "Applying specialization to process models", Conference on Organizational Computing Systems, Tools", 1995, pp. 290-301.

[7] Klein M. and Konig-Ries B., "A process and a tool for creating service descriptions based on DAML-S", 4th VLDB Workshop on Technologies for E-Services (TES'03), Berlin, Germany, 8 September 2003.

[8] Muth,P., Wodtke,D., Weissenfels,J., Dittrich,A. and Weikum,G., "From centralized workflow specification to distributed workflow execution", Journal of Intelligent Information System, vol. 10, no. 2, 1998.

[9] Paolucci M. and Sycara K., "Autonomous semantic web services", IEEE Internet Computing, vol. 07, no. 5, pp. 34-41, September/October 2003.

[10] Peter Mika and Daniel Oberle and Aldo Gangemi and Marta Sabou, "Foundations for service ontologies: aligning OWL-S to code", 13th International Conference on World Wide Web, New York, USA, 2004, pp. 563-572.

[11] S.K.Ong and EnyaKong Tang, "Service description and composition using ontologies", 23rd IASTED International Conference on Artificial Intelligent and Applications (AIA2005), Innsbruck, Austria, Feb 2005.

[12] Thomas W.Malone and Kevin Crowston, Jintae Lee, Brian Pentland, Chrysanthos Dellarocas, George Wyner, John Quimby, Charles S.Osborn, Abraham Bernstein, George Herman, Mark Klein, and Elissa O Donnell, "Tools for inventing organizations: towards a handbook of organizational process", Management Science, vol. 45, no. 3, pp. 425-443, March 1999.

[13] Yuri A. Tijerino, David W. Embley, Deryle W. Lonsdale, and George Nagy. "Ontology Generation from Tables," Wise, vol. 00, pp. 242, Fourth 2003.

Service Description and Composition Using Ontologies

Ong Siew Kin and Tang Enya Kong
Computer Aided Translation Unit
University Science Malaysia
{skin, enyakong}@cs.usm.my

Abstract

Electronic marketplace consists of many competing companies who supply each other with web services. The rapid growth of the number of Web services makes finding, selecting and combining them a desirable but challenging task. In this paper, we are proposing the Ontology Web Language (OWL) to capture the content knowledge and Web services. The proposed architecture is part of the mFinance project that utilizing ontologies to describe knowledge and services. We are prototyping a service aggregator that serves as a service repository. It keeps descriptions of services that are accessed by the requester agent and advertised by the service provider. We show how service description, selection and composition are possible using ontologies.

Keyword: Knowledge Representation, Ontology, Semantic Web Service, E-Commerce

1. Introduction

Electronic marketplace consists of many competing companies who supply each other with web services. The rapid growth of the number of Web services makes finding, selecting and combining them a desirable but challenging task. There is a need to standardize the descriptions of services to allow interoperability. Standards such as Soap Object Access Protocol (SOAP) [1] and Web Service Description Language (WSDL) [2] which are based on XML provide descriptions of messages transport mechanisms and descriptions of the interface used by each service. However, neither SOAP nor WSDL facilitates the automatic location and composition of services on the basis of their capabilities. The emerging XML-based standard Universal Description, Discovery and Integration (UDDI) [3], which provides a registry of businesses and Web services, also does not help in capability-based searching and composition because it lacks the explicit description of capabilities. These proposed standards require programmers to hardcode the Web services with information about their interaction partners, the messages to exchange and the interpretation of the messages that they receive. The result is a set of rigid Web services that cannot reconfigure dynamically to adapt to changes without direct human intervention. Such hard-coded Web services are limited to work with a definite set of providers and unable to reconfigure when a new provider comes on line. This limitation has pushed the growing of interests in the use of ontologies as a means to facilitate interoperability. The interoperability is achieved through the explicit modeling of the intended meaning of concepts used in service descriptions between different information sources, software components and service providing softwares.

Service discovery involves the matching of two service descriptions based on the properties that each service has. A service advertisement is considered as a match for a request when the advertisement is sufficiently similar to the request. The sufficiently similar revolves around the concept of subsumption which is provided by ontology. In this paper, we are proposing the Ontology Web Language (OWL) [4] to capture the content knowledge. Initiated by the Ontology Web language – Service (OWL-S), we describe the services using Protégé 2000. The ontologies will be kept in a service aggregator that serves as a service repository. It keeps descriptions of services that are accessed by the requester agent and advertised by the service provider. It also serves as the platform to perform service composition.

The rest of the paper is outlined as follows. In section 2, we discuss on the requirements needed in service description language. Then, we introduce OWL-S as the candidate service ontology in section 3. In section 4, we show how to describe and compose services based on the service descriptions we have created. We mention some related works in section 5 and finally we conclude this paper in section 6.

2. Description Language Requirements

To facilitate service discovery and composition, the service description language should meet the following requirements:

1. Sufficiently rich to capture as little or as much of the capability of the service as needed
2. Flexible and extensible
3. Support arbitrary granularity

The description should include the following information:

- The type of problem, input and output data and type
- Semantic supported (example UNSPSC, RosettaNet, etc.)
- Logistical information or functionality attributes (example cost, quality of service, etc.)
- Service binding description

Once the services are merged together to form a uniform space, ownership and tractability of service instances might need to be distinguished. The data owner might wish to limit the access control to their data or wish to remove their data easily. Secondly, the service consumer might wish to track back to the service sources. Both requirements can be easily solved by augmenting the service description with a provider profile.

The service descriptions should be classified to create an index to the descriptions. Besides, classification helps to impose the filtering mechanism and partial or imprecise match. The same service description can be classified differently depending on the view taken. This is called 'multi-axial classification'. Service classification can base on the service semantics or operations, functionality or the input and output interface, or the organization that provides them.

3. OWL-S

OWL-S [5] is an OWL ontology which supplies the service provider a core set of markup language constructs for describing the properties and capabilities of service. It has the same goals as the previous DAML-S to describe services in an unambiguous, computer-interpretable form. The OWL-S ontological description aims to allow automatic discovery, invocation, interoperability, composition of new services, verification of services and execution monitoring.

OWL-S adds rich typing and class information that we can use to describe and constrain the range of Web service capabilities. Such language enables the grouping of like services and like data types into taxonomic hierarchies, together with rich definitions of the relationships and constraints between classes and their instances. Process inputs and outputs are named and typed using either OWL concepts (classes or properties) or data types that XML Schema provides.

In OWL-S, a service is defined by three concepts (see figure 1): a profile, a model and a grounding.

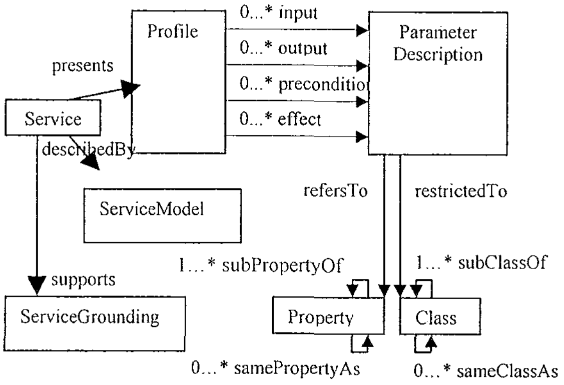


Figure 1 OWL-S model

OWL-S allows for the description of a Web service in terms of a Profile, which tells "what the service does", a Process Model, which tells "how the service works", and a Grounding, which tells "how to access the service". The Profile and Process Model are considered to be abstract specifications, in the sense that they do not specify the details of particular message formats, protocols, and network addresses by which a Web service is instantiated. The role of the grounding is to provide these more concrete details [6]. Generally speaking, the Profile provides the information needed for an agent to discover a service. Taken together, the Model and Grounding objects associated with a service provide enough information for an agent to make use of a service. For the service composition, we will focus on the Profile to match the input and output descriptions.

A service profile provides a high-level description of a service and its provider. There are three types of information captured in a profile: a human readable description of the service, a specification of the functionalities that are provided by the services, and the functional attributes. Human readable description captures information about the service, such as its provenance, service name and text summary. Functionality description specifies what the service provides and the conditions that have to be satisfied for a successful result. The service is represented by input and output properties of the profile. Preconditions show one or more logical conditions that should be satisfied prior to the service being requested. These conditions should have associated explicit effects that may occur of a service. Since different services might only applicable to a specific audience, there are other aspects of services that should be captured. Some functional attributes are outlined below:

- Geographic Radius – refers to the geographical scope of the service.
- Degree of Quality – provides qualifications for the service.
- Service Parameter – an expanded list of properties that may accompany a profile description.
- Service Type – refers to a high level classification of the services.

The attributes outlined above are not complete. Additional attributes can be added depending on the system requirements. In the next section, we only show the attributes relevant for our service selection and composition.

4. Service Aggregator Using Ontologies

A suite of ontologies are created to provide the service classification, a shared vocabulary for expressing service descriptions as well as reasoning processes to both manage the coherency of the classifications and the descriptions when they are created, and the service discovery, matchmaking and composition when they are deployed. Each service description refers to a number of ontologies that define the domains of parameters like input, output, precondition, effect and a number of constraints between the parameters. There are two main types of ontology involved in our prototype system, the domain ontology and the service ontology (see figure 2). The domain ontology will capture knowledge of interests (Weather Ontology, Location Ontology, etc.) while the service ontology describes services that refereeing the domain ontology. The scope of the ontology is determined by requirement to represent the service descriptions.

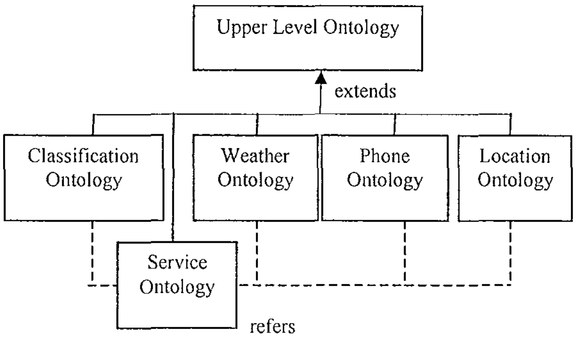


Figure 2 Ontologies for service description

The aggregator will interpret the service capabilities advertised by the provider and return the best matching services. While finding the best match, the aggregator could compose new service workflow if the single service could not fulfill the service request.

Our service aggregator model is shown in figure 3. Concept Service will be used to classify services. It provides service on some resources. Each service is presented by many profiles, which are supplied by providers. The profile supports a grounding that specifies how the service should be invoked.

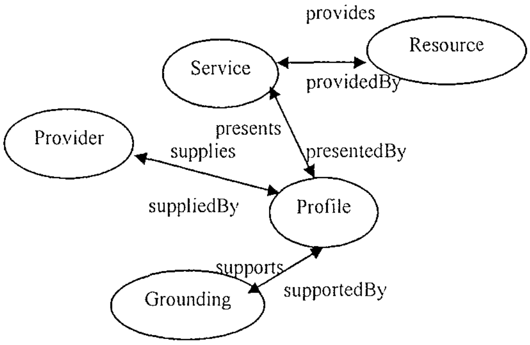


Figure 3 Service Aggregator Model

4.1. Service Description and Composition

Our prototype system is build using the Protégé 2000 toolkit [7]. The system allows service provider to markup up their services using the OWL ontology. Figure 4 and table 1 shows some services that are advertised in the aggregator. There are five service profiles (LocationProfile01, LocationProfile02, LocationProfile03, WeatherProfile01 and WeatherProfile02) registered under the GetLocation and GetWeather Service, each requires different inputs, outputs, preconditions and effects as listed in table 1.

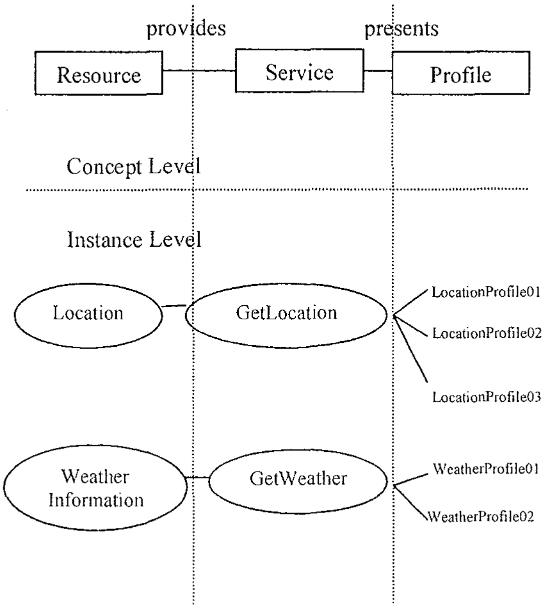


Figure 4. GetLocation and GetWeather services

Table 1. Service profile descriptions

LocationProfile01	
Has Input	MSISDNNUMBER DIAMETERVALUE
Has Output	LOCNAME
Precondition	<owl:Class rdf:ID="CondPhone"> <rdfs:subClassOf rdf:resource="#Condition"/> <rdfs:subClassOf> <owl:Restriction> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> <owl:onProperty> <owl:DatatypeProperty rdf:about="#phoneNumber"/> </owl:onProperty> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
Effect	<owl:Class rdf:ID="CondGeoPosition"> <rdfs:subClassOf> <owl:Restriction> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> <owl:onProperty> <owl:DatatypeProperty rdf:about="#latitude"/> </owl:onProperty> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf rdf:resource="#Condition"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty> <owl:DatatypeProperty rdf:about="#longtitude"/> </owl:onProperty> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>

LocationProfile02	
Has Input	PHONENUMBER
Has Output	LOCNAME
Precondition	<owl:Class rdf:ID="CondName"> <rdfs:subClassOf rdf:resource="#Condition"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty> <owl:DatatypeProperty rdf:about="#name"/> </owl:onProperty> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
Effect	<owl:Class rdf:ID="CondGeoPosition"> <rdfs:subClassOf> <owl:Restriction> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> <owl:onProperty> <owl:DatatypeProperty rdf:about="#latitude"/> </owl:onProperty> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf rdf:resource="#Condition"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty> <owl:DatatypeProperty rdf:about="#longtitude"/> </owl:onProperty> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>

LocationProfile03	
Has Input	LOCNAME
Has output	LONGITUDE LATITUDE
Precondition	<owl:Class rdf:ID="CondGeoPosition"> <rdfs:subClassOf> <owl:Restriction> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> <owl:onProperty> <owl:DatatypeProperty rdf:about="#latitude"/> </owl:onProperty> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf rdf:resource="#Condition"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty> <owl:DatatypeProperty rdf:about="#longtitude"/> </owl:onProperty> <owl:cardinality

	rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
Effect	<owl:Class rdf:ID="CondName"> <rdfs:subClassOf rdf:resource="#Condition"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty> <owl:DatatypeProperty rdf:about="#name"/> </owl:onProperty> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>

WeatherProfile01

Has Input	LOCNAME
Has Output	MAXDEGREE MINDEGREE
Precondition	<owl:Class rdf:ID="CondName"> <rdfs:subClassOf rdf:resource="#Condition"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty> <owl:DatatypeProperty rdf:about="#name"/> </owl:onProperty> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
Effect	<owl:Class rdf:ID="CondTemperature"> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty> <owl:DatatypeProperty rdf:about="#maxDegreeC"/> </owl:onProperty> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf> <owl:Restriction> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> <owl:onProperty> <owl:DatatypeProperty rdf:about="#minDegreeC"/> </owl:onProperty> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf> <owl:Class rdf:ID="Condition"/> </rdfs:subClassOf> </owl:Class>

WeatherProfile02

Has Input	LATITUDE LONGITUDE
Has Output	SKYCONDITION
Precondition	<owl:Class rdf:ID="CondGeoPosition"> <rdfs:subClassOf> <owl:Restriction> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> <owl:onProperty> <owl:DatatypeProperty rdf:about="#latitude"/> </owl:onProperty> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf rdf:resource="#Condition"/> <rdfs:subClassOf> <owl:Restriction> <owl:onProperty> <owl:DatatypeProperty rdf:about="#longtitude"/> </owl:onProperty> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> </owl:Restriction> </rdfs:subClassOf> </owl:Class>
Effect	<owl:Class rdf:ID="CondSkyCondition"> <rdfs:subClassOf> <owl:Restriction> <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#int" >1</owl:cardinality> <owl:onProperty> <owl:DatatypeProperty rdf:about="#skyCondition"/> </owl:onProperty> </owl:Restriction> </rdfs:subClassOf> <rdfs:subClassOf rdf:resource="#Condition"/> </owl:Class>

Possible service requests for weather information that involves service matching and composition are listed below:

1. Service request 1
   Input: location name
   Output: Weather information
   Precondition: CondName
   Effect: CondTemperature, CondSkyCondition

   The reasoner derives that the output set required are MAXDEGREE, MINDEGREE and SKYCONDITION based on the attributes defined for concept Weather. The effects needed are CondTemperature and CondSkyCondition. The composer decides that both WeatherProfile01 and WeatheProfile02 are needed to fulfill the output and effect set. The WeatherProfile01 is selected because the input is matched with the LOCNAME and the precondition holds. However, to get the SKYCONDITION, the WeatheProfile02 requires LATITUDE and LONGITUDE as inputs (or CondGeoPosition as the precondition). The composer will match the new input and precondition set with the output and effect sets of registered services. LocationProfile03 is selected since the input, output, precondition and effect sets are matched. So, the new service composition will be LocationProfile03 → WeatherProfile02 and the WeatherProfile01 as shown in the figure 5.

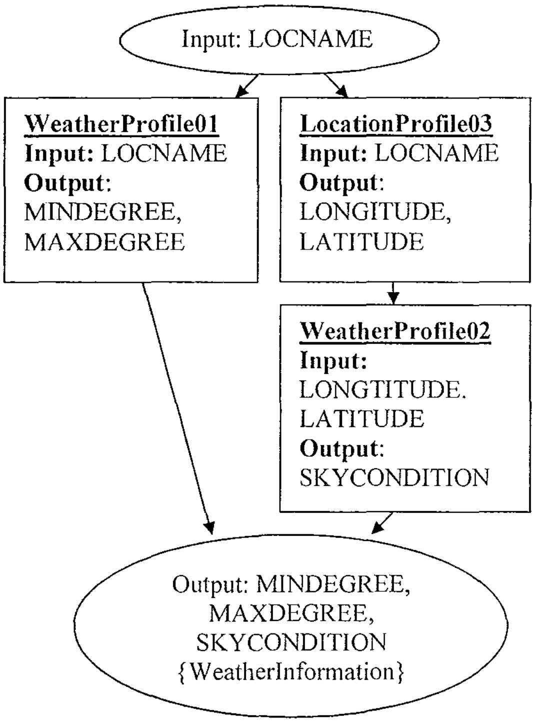

Input: LOCNAME
WeatherProfile01
Input: LOCNAME
Output: MINDEGREE, MAXDEGREE
LocationProfile03
Input: LOCNAME
Output: LONGITUDE, LATITUDE
WeatherProfile02
Input: LONGTITUDE. LATITUDE
Output: SKYCONDITION
Output: MINDEGREE, MAXDEGREE, SKYCONDITION {WeatherInformation}

Figure 5. Service composition for Input: location name; output: Weather information

2. Service request 2
   Input: phone number
   Output: maximum and minimum degree of the day
   Precondition: CondPhone
   Effect: CondTemperature

   The service composition will be LocationProfle02 → WeatherProfile01

3. Service request 3
   Input: MSISDN number, diameter
   Output: Weather information
   Precondition: CondPhone
   Effect: CondTemperature, CondSkyCondition

   The service composition will be LocationProfle01 → WeatherProfile01 and LocationProfile01 → LocationProfile03 → WeatherProfile02

5. Related Work

AgentCities [8] is an example that explores the use of agents in a distributed environment. It builds a network of agent platforms that represent the virtual cities and populates them with services such as dining, movies, etc... There has been a number of different domain and service ontologies. The domain ontologies represent the terms for restaurant and movies while the service ontology captures the terms used in the actual services. Some key issues from this work are the use and integration of multiple separate ontologies, potentially multiple ontologies for the same domain or service and the use of ontologies in describing the services.

The DAML-S Matchmaker [9] is a system to augment current UDDI architecture with semantic service descriptions. The Matchmaker aims to improve the discovery process by allowing location of services based on their capabilities. The basic idea used in the matchmaker is making use of the subsumption relation between the classes to find flexible matching beyond the capabilities of the UDDI. After locating the services, the DAML-S Virtue Machine [10], allows the autonomous interaction and invocation of Web services based only on the DAML-S specification with no need of the prior hard-coding that would be needed by the Web services specifications such as WSDL or Business Process Execution language for Web Services (BPEL4WS).

6. Conclusion

In this paper, we have introduced a service aggregator that could act as a composer agent. It enables the service provider to describe the service semantically, helps selecting and composing the services required using ontologies. Our approach utilizes the service profile and matches only the precondition and effect sets. We are planning to do matching on the functional attributes to allow more flexible composition. This paper does not discuss the service Model and Grounding issues in details. To allow execution of service, it is necessary to analyze the Model and Grounding to come up with a concrete description for service invocation. Our current implementation has assumed that all services are atomic and the service invocation can be as simple as HTTP and internal functional call, or as complex as SOAP invocation. Our future works will look into the service Model and Grounding issues.

7. Acknowledge

This work is part of the m-Finance project funded by iNavigate Sdn. Bhd. in collaboration with University Science Malaysia.

References

[1] SOAP http://www.daml.org/services/owl-s/1.0/

[2] WSDL http://www.w3c.org/TR/2001/NOTE-wsdl-0010315

[3] UDDI "UDDI Technical White Paper", 2000, http://www.uddi.org/whitepapers.html

[4] OWL Web Ontology Language Reference http://www.w3c.org/TR/2003/CR-osl-ref-20030818/

[5] The OWL Service Coalition: Semantic Markup for Web Services (OWL-S): http://www.daml.org/services/owl-s/1.0

[6] Paolucci M., Sycara K., and Kawamura T., "Delivering Semantic Web Services", In Proceedings of the 12th International Conference on World Wide Web. ACM Press, 2003

[7] M.A.Musen, R.W.Fergerson, W.E. grosso, N.F.Noy, M.Crubezy, & J.H.Gennari. "Component-Based Support for Building Knowledge-Acquisition Systems". In Conference on Intelligent Information Processing (IIP 2000) of the International Federation for Information Processing World Computer Congress (WCC 2000), Beijing, 2000

[8] Agent cities, http://www.agentcities.org

[9] Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara, "Semantic Matching of Web Services Capabilities", *The First International Semantic Web Conference (ISWC)*, Sardinia (Italy), June, 2002.

[10] Massimo Paolucci, Anupriya Ankolekar, Naveen Srinivasan and Katia Sycara, "The DAML-S Virtual Machine", In Proceedings of the Second International Semantic Web Conference (ISWC), 2003, Sandial Island, Fl, USA, October 2003, pp 290-305.

[11] Mikko Laukkanen and Heikki Helin, "Composing Workflows of Semantic Web Services", In *AAMAS Workshop on Web Services and Agent-Based Engineering*, 2003.

Information Integration and Semantic Interoperability Support in Ontologies

Short term grant technical report

By

Gian Chand Sodhy
Unit Terjemahan Melalui Komputer, Pusat Pengajian Sains Komputer, USM, Penang
Sodhy@cs.usm.my

ABSTRACT

The vision of Semantic Web Services (SWS) is to create dynamically interoperating nodes on the web. To help realising the SWS, our work conducted under this short term grant proposed a model to organise information integration and semantic interoperability support in ontologies. Besides, we propose a method to transform existing web sites into semantic web services. We achieve this by defining our domain and state ontologies, analysing the navigational behaviors of web sites using state-charts and instantiating semantic web services by annotating them with state ontologies and domain ontologies.

KEYWORDS

Semantic Web, Ontology, Semantic Web Services

1. Introduction

The vision of SWS is to create dynamically interoperating nodes on the web. In supporting of this effort, technologies from Web Services and Semantic Web have been tailored towards this vision. These include XML standard as the unifying factors, SOAP as the message passing definition, WSDL as the description language for service interface, and BPEL4WS as the mechanisms for describing interaction among several web services. On the other hands, the Semantic Web offers the ontological concepts in annotating web service capabilities as well as interactions. Therefore, SWS can be seen as using ontologies and semantically annotated service descriptions to automate the fulfillment of tasks and transactions [9]. Our work is in the same direction as the proposed SWS, we propose a model to organize the semantic web services and we aim to fasten the adoption of SWS by transforming current web sites (offering certain services through HTML forms) into semantically annotated web services. Section two of the report outlines our proposed SWS organizing model. Section three describes the requirements for web service annotation and section four briefs the creation of our domain ontologies. The process of wrapping web sites into semantic web services is outlined in section five before we conclude our report.

2. WEB SERVICES ORGANIZATION

Inspired by the work [2] on layered service ontology, we have taken some aspects of the layering to cater for two-dimensional view on web services. Upper service ontology provides the basic structure for web service description. DAML-S [1] and its successor OWL-S [10] are potential candidates. These ontologies provide a set of basic markup primitives that could be used to describe different attributes of web services. Our model organizes semantic web services by domain service ontology. Community from the same domain can usually reach consensus easily on matters like vocabularies (domain ontologies) used to communicate, security policies and communication protocols. In our work, we assume that different parties from the travel domain have formed a community and are initiating the semantic web services organisation. There are two different views in analysing processes[12]: parts of a process and types of a process. In our approach, we view web services as processes and organise them by part and by type. Organising the services in this way facilitates the discovery (browsing and searching) and composition of semantic web services [11]. The third element in the model is made up of many domain specific ontologies and state ontologies. This group of ontologies provides common vocabularies for describing the services.

3. WEB SERVICES DESCRIPTION

There are many ways to describe a web service. The quality of service description will affect directly its applicability. By just describing the service name and its provider, we can do a simple keyword-search based on these criteria. By adding semantic to the service description (marking with ontology terms), searching on the services can now be based on semantic and not just pure keywords. Annotating service with attributes like input, output, precondition and effect of the service will enable a functional search and simple composition. To allow the composing agent to automatically compose and further execute the service, the internal behaviors and the binding information of a service must be captured. Therefore, we realize that specification of semantic web services which aim at automatic discovery, composition, execution and monitoring should include semantic information about both data flow (functionalities) and control flow (behaviors). Besides describing the basic interface of web services as proposed by W3C in DAML-S, our work use state charts [8] as the workflow specification language. We use DAML-S in this project although OWL-S is the newest recommendation. In the future, we will use OWL-S instead.

4. DOMAIN ONTOLOGY CREATION

In Semantic Web, we need domain ontologies that organise the most relevant concepts, their relationships, and instances. In SWS, ontologies provide a shared conceptualization for interpreting semantic markup of web services. It enables services across the web to use the same terminologies to interpret each other's message and behavior description.

In our work, we need ontologies in domain relevant to travel as well as state ontologies that describe the various stages of transaction. In particular, we need to describe service input, output, pre-condition and effect with ontology vocabularies. The state ontology is created after analysing the web sites by collecting the states from the state-charts. The remaining text in this section refers to creation of domain ontologies. Creating ontological descriptions for real-world information is nontrivial. There are some available public domain ontologies and communities in electronic business have created several standards [5] (product, catalog and document structure standard) in effort to promote information integration. An example of standards in identifying e-commerce products, United Nation Standard Products and Services Codes (UN/SPSC) defines a hierarchical classification with five levels with each level containing a two-character numerical value and a textual description. While creating ontologies for use in our work, we reuse some terminologies from the UN/SPSC and enhance it further with WORDNET [3]. There are several issues to be considered for a successful ontology such as:

- Industry specific need have to be taken into account. Essential attributes and values of a product highly differ per industry.
- Country specific needs and regulations
- Should be broadly supported by standardisation bodies.
- Needs broad industry support (from both businesses, marketplaces, and e-commerce software vendors)

By considering these issues, our SWS model categorises semantic web services by domain community to ease the creation and utilisation of ontologies.

5. INSTANTIATING THE SEMANTIC WEB SERVICES

Before instantiating semantic web services, we need to acquire the generic service description from the current web sites. We do this by analysing the navigation behavior of web sites using state-charts. Various states identifying in the state-charts makes up the content for state ontology. Generalization technique [6] is applied on the collected state-charts to create a generic service behavior for similar services. These generic service behaviors will be the attributes of services as defines in our service category ontology.

After getting the generic service description, we populate the service model with real semantic web services using the wrapping technique. Each web site is categorised under a certain service category, which imports several states from the state ontology. Service instance is further marked up using terminologies from the domain ontologies to describe its required input and output. Each corresponding state-charts will be transformed into workflow language (BPEL4WS) and its pre-state and post-state will become its pre-condition and effect respectively. State-chart provides important information about service precondition and effect but not input and output information. Input and output information could be obtained by analysing web elements such as web form elements and returned result. Each event in the state-chart is the function in our service description (Notice that we add two events into every service description (init() and end() to denote the start and the end of service flow).

6. CONCLUSION

To realise the vision of SWS, there is a need for an organising model. In this project , we proposed and implemented a model that facilitates the creation, discovery and composition of semantic web services. By focusing on domain classification, agreement on ontologies can be easily achieved and commitment on the ontologies is better secured. We also contributed in fastening the creation of semantic web services. By using state-charts as the analysis tool, we successfully derived the generic service description that captures the behavior attributes of services. Our future work will further evaluate how our model can facilitate the composition of semantic web services.

REFERENCES

[1] Anupriya Ankolekar, Mark Burstein, Jerry Hobbs J., et al. "DAML-S: web service description for semantic web", First International Semantic Web Conference (ISWC02), 2002.

[2] B. J. Wielinga, A. Th. Schreiber, J. Wielemaker and J. A. C.Sandberg, K-CAP 2001: Proceedings of the international conference on Knowledge Capture, NY: ACM Press, 2001, Chapter From Thesaurus to Ontology, pp. 194-201.

[3] Christiane Fellbaum, Wordnet: an electronic lexical database (language, speech, and Communication, The MIT Press, 1998.

[4] Davalcu, H., Vadrevu, S., Nagarajan, S. and Ramakrishnan, I.V., IEEE Intelligent Systems and Their Applications , IEEE Intelligent System, 2003, Chapter OntoMiner: bootstrapping and populating ontologies from domain-specific Web sites, pp. 24-33.

[5] Fairchild, A.M., and de Vuyst, B., "Coding Standards Benefiting Product and Service Information in E-commerce", 35th Annual Hawaii International Conference on System Sciences (HICSS'02), 7-10th Jan 2002. vol. 8, pp. 3201-3208.

[6] George M.Wyner and Jiantae Lee, "Applying specialization to process models", Conference on Organizational Computing Systems, Tools", 1995, pp. 290-301.

[7] Klein M. and Konig-Ries B., "A process and a tool for creating service descriptions based on DAML-S", 4th VLDB Workshop on Technologies for E-Services (TES'03), Berlin, Germany, 8 September 2003.

[8] Muth,P., Wodtke,D., Weissenfels,J., Dittrich,A. and Weikum,G., "From centralized workflow specification to distributed workflow execution", Journal of Intelligent Information System, vol. 10, no.2, 1998.

[9] Paolucci M. and Sycara K., "Autonomous semantic web services", IEEE Internet Computing,vol. 07, no. 5, pp. 34-41, September/October 2003.

[10] Peter Mika and Daniel Oberle and Aldo Gangemi and Marta Sabou, "Foundations for service ontologies: aligning OWL-S to code", 13th International Conference on World Wide Web, New York, USA, 2004, pp. 563-572.

11] S.K.Ong and EnyaKong Tang, "Service description and composition using ontologies", 23rd IASTED International Conference on Artificial Intelligent and Applications (AIA2005), Innsbruck, Austria, Feb 2005.

12] Thomas W.Malone and Kevin Crowston, Jintae Lee, Brian Pentland, Chrysanthos Dellarocas, George Wyner, John Quimby, Charles S.Osborn, Abraham Bernstein, George Herman, Mark Klein, and Elissa O Donnell, "Tools for inventing organizations: towards handbook of organizational process", Management Science, vol. 45, no. 3, pp. 425-443, March 1999.

13] Yuri A. Tijerino, David W. Embley, Deryle W. Lonsdale, and George Nagy. "Ontology Generation from Tables," Wise, vol. 00, pp. 242, Fourth 2003.