Domain Modelling for Ubiquitous Computing Applications

Anthony Harrington and Vinny Cahill
Distributed Systems Group
School of Computer Science and Statistics
Trinity College Dublin

E-mail: Firstname.Lastname@cs.tcd.ie

Abstract

Many Ubiquitous computing applications can be considered as planning and acting problems in environments characterised by uncertainty and partial observability. Such systems rely on sensor data for information about their environment and use stochastic or probabilistic reasoning algorithms to infer system state from sensor data.

We propose a domain modelling technique that characterises the sensor and actuator infrastructure and the set of system states in Ubicomp application domains. We capture the location and geometry of all domain model elements and use these spatial properties to tailor system state determination to reflect the quality and spread of the sensor and actuator platform.

The domain model is used to support the development of Ubicomp applications and allows us calculate the degree of observability that exists over the state space in the environment. The degree of observability over the state space can act as an input into determining the state inference and action selection algorithms used in Ubicomp systems.

In this paper we present the design of our Ubicomp domain model. We show that the proposed model contains information necessary to support Ubicomp application development. The expressiveness of the proposed design has been tested by building a model of an Urban Traffic Control application for Dublin city.

1 Introduction

Many Ubiquitous computing (Ubicomp) applications can be considered as real world planning and acting problems. Canonical examples of such applications are the control of transportation infrastructures, activities such as region-wide pollution monitoring and emergency service management.

Ubicomp applications can be modelled as a set of system goals, actions that the system is capable of taking and the decisions that must be taken to achieve the desired outcome or state. The set of actions associated with Ubicomp applications is bounded by the functionality of the actuator infrastructure deployed in the environment or application domain. Ubicomp applications are also optimisation problems in so far as they attempt to maximise or minimise some property of the environment.

Partial observability is a challenging characteristic that real world environments present as application domains and stems from the dependence on sensor data for state determination with the attendant possibility of error or partial coverage in the sensor readings. The sensor platform sits between the application and the environment thereby precluding full observability and global knowledge of system state. The uncertainty in state determination may require the use of stochastic or probabilistic reasoning algorithms that infer system state from sensor data [15].

Ubicomp applications acting in uncertain environments may be modelled using classical planning techniques that have been extended to explicitly model probabilistic state transitions and observations [4]. These applications use control or optimisation algorithms for choosing actions for changing the state of a system so that it moves towards its desired goal state.

An important result from optimisation theory are the *No Free Lunch* theorems [17] that state that there is no one single optimisation strategy or algorithm that is optimum over the set of all possible problems. These theorems imply that the choice of optimisation strategy in Ubicomp applications should reflect the nature of the environment and the problem structure.

In a complex real world domain, application state-space sizes may become intractably large for search and exploration algorithms [12]. Domain specific knowledge can be used as a heuristic to bound the state-search space when de-

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

vising a solution and should be exploited where available [4] [11].

This analysis of Ubicomp applications indicates that properties of the application domain are very relevant when building such systems. We have developed a domain modelling process to support the profiling of Ubicomp domains and that can be used to support Ubicomp application development.

The proposed domain model is independent of an individual Ubicomp application and supports the development of multiple applications in a common domain. The principal requirement of the Ubicomp domain model is that it should contain the information necessary to model a Ubicomp application under development. The multi-layered domain model characterises the sensor and actuator infrastructure in the domain and the set of system states over which the application is defined. In a partially observable environment the ability to determine the system state or the impact or efficacy of actions is dependent on the quality and spread of sensor infrastructure. Therefore the domain model must also support the development of an observation model that allows us to reason about the state of the system in uncertain environments.

To meet these requirements our domain model uses Geographic Information Systems (GIS) support to allow state determination to reflect sensor and actuator position and spread. The domain model provides a uniform and scalable method of generating observation models that measure the quality of the sensor infrastructure and allows us to determine where noise tolerant planning and optimisation strategies are appropriate.

The Ubicomp domain model is used to construct an application specific planning model. Our planning model is an application specific set of information extracted from a domain model and is a direct input into the development process.

This paper is structured as follows. In section II we formalise the notion of a planning model for Ubicomp systems and present the architecture of our domain model. In Section III we present the design of our domain model. In Section IV we present a case study of applying the domain model to the development of an Urban Traffic Control (UTC) system in Dublin city and show how to construct a planning model from our domain model.

2 Domain Modelling and Planning for Ubicomp

In this section we formalise the notion of a Ubicomp planning model that encodes our knowledge of an application. We also present the architecture of our multi-layered domain model that stores the information necessary to construct a Ubicomp planning model.

The planning model is suitable for use in uncertain environments and is based on the standard model of state-transition systems [4]. It is a five-tuple $\sum = (S, A, E, \gamma, O)$ where:

- $S = \{s_1, s_2, ..\}$ is a finite set of states;
- $A = \{a_1, a_2, ..\}$ is a finite set of actions;
- $E = \{e_1, e_2, ..\}$ is a finite set of events;
- $\gamma : S \times A \times E \rightarrow 2^s$ represents a state-transition function that may be non-deterministic to reflect a stochastic environment. If γ is non-deterministic and P is a probability distribution that represents the transition probabilities for each action in each state then $P_a(s^{'}|s)$ is the probability that if we execute an action a in state s, then a will lead to state $s^{'}$.
- $O = \{o_1, o_2, ..\}$ is a finite set of observations with probabilities $P_a(s|o)$, for any $a \in A, s \in S$ and $o \in O$. $P_a(s|o)$ represents the probability of observing s after executing action a and receiving observation o. Observations are produced by the sensor infrastructure and identify the set of possible states the system might be in at each time step.

Ubicomp applications may be implemented using various components of such a planning model. The choice of optimisation technique applied in an application may be dependant on the information available to the developer. Consider the problem of optimising traffic flow in a city-wide region. Such an application if constructed as a Partially Observable Markov Decision Process (POMDP) would reason over observed belief states rather than the actual system state space [2]. If the application was constructed using a Reinforcement Learning algorithm the application would begin with no state-transition system and would attempt to learn γ using exploratory actions [1] [3]. Finally, if such a system were engineered using a Genetic Algorithm it would use a simulation of the system coupled with randomly selected state-space combinations to search for an acceptable solution to the problem [8]. The information required when building an application varies according to the choice of optimisation algorithm employed by the designer.

Our first step when designing Ubicomp systems is to build a domain model that characterises important properties of the domain. This domain model is used to construct a planning model specific to the application under development. The planning model is independent of the choice of optimisation algorithm that will be used. The goal is to use the information contained in the planning model as an input when selecting an appropriate optimisation algorithm during application development.

It is also likely that a single domain model will be useful to many applications. Consider the problem of emergency

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

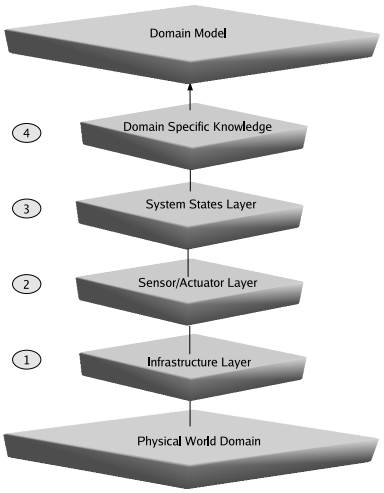

Figure 1. Layered Domain Model

service vehicle priority at traffic junctions. A domain model developed for the UTC domain will be useful for both vehicle priority and flow optimisation applications. With this consideration in mind, the domain model has been designed to be extensible and support many data formats. It is intended to provide a reusable basis for developing multiple applications in a domain.

2.1 Domain Model Architecture

The domain model is shown in Figure 1. There are four stages in the domain modelling process. The output of each stage is reflected in a layer of the domain model.

Layer 1 specifies the physical artifacts that exist in the application domain. The choice of relevant infrastructural elements is domain and application dependent. Physical artifacts relevant to the transportation domain include the location and geometry of transport network junctions and links, whereas office equipment and layout would be relevant artifacts in a smart office application.

Layer 2 specifies the functionality and spread of the sensor and actuator deployment in the application domain. This information contains the set of sensor states and actuator actions. The quality and spread of the sensor data will be used to determine the degree of observability over the system state space.

Layer 3 provides a higher-level partitioning of the domain into system states that is more reflective of the application goals. Such states are typically composites of multiple sensor states.

Layer 4 of the domain model represents domain specific knowledge. Domain specific knowledge can act as a heuristic guide to a planning model for bounding the state-space search in uncertain domains [4]. Domain knowledge may also be encoded in expert or rule based systems.

If no domain specific knowledge is available for an application then this layer of the model will be empty. In this scenario, a learning or stochastic search algorithm could be applied to optimise the system. Our approach is guided by the belief that if domain knowledge is present then we should utilise it. The use of elements in layers 1 to 3 is independent of the presence or absence of domain knowledge in layer 4.

3 Domain Model Design

The principal requirements for the domain model design are that the model should contain the information necessary for constructing a Ubicomp planning model as described in Section II. It should also support the generation of observation models that allow us to reason about the state of the system in uncertain environments.

The domain model classes are designed to map directly into the components of the planning model presented in Section II. Model elements are tagged with spatial attributes representing their location and geometry. Spatial attributes are used to match sensor and actuator positions to system states across regions of the domain and thereby determine the degree of observability over the set of system states.

3.1 Domain Model Abstractions

The domain model been designed using an object-oriented methodology. The Model Element is the base class from which all layer elements are derived. It contains two attributes: the *type* and *dataSourceId* that identify respectively the domain model layer to which the element belongs and the data format used to model the element. Layers 1 and 2 support a number of standard GIS data formats. Layers 3 and 4 use XML schema to represent system states and domain specific knowledge.

The Location Reference class contains a *referenceSystemId* attribute that identifies the coordinate reference system used. Explicitly specifying the location referencing system used for each element allows model elements from separate layers to be overlayed and matched against each other. It also contains a *location* attribute that specifies the physical location and geometry of elements in the domain model.

3.2 Domain Model Layers

3.2.1 Infrastructure Element Layer

Layer 1 specifies the physical artifacts that exist in the application domain. This layer supports numerous data formats including the Geography Markup Language (GML)

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

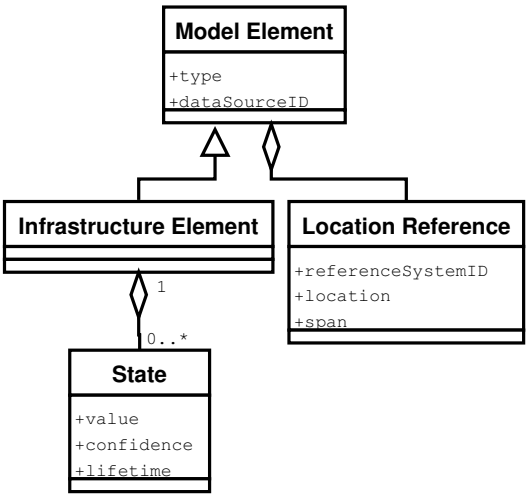

Figure 2. Infrastructure Layer

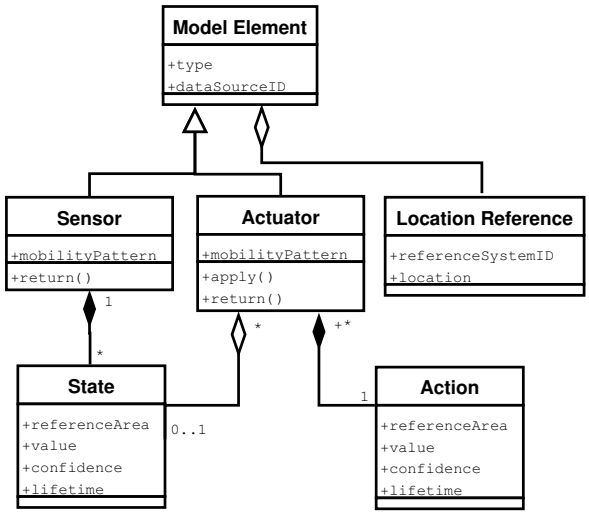

Figure 3. Sensor Actuator Layer

[1], Simple Features Specification for SQL (SFS)[2] and ESRI Shapefiles[3]. The design of this layer is shown in Figure 2. In an Urban Traffic Control (UTC) scenario, infrastructure elements include road junctions and links. The spatial attribute of a road link element may be specified as a polygon object using a standard coordinate reference system. Likewise the spatial attributes of a traffic light object may be represented using a point object.

3.2.2 Sensor/Actuator Layer

Layer 2 models the set of sensor states and actuator actions. The spatial attribute of a sensor includes both its physical location and sensing range. The design of this layer is shown in Figure 3.

The Sensor and Actuator elements derive from the Model Element. Both classes contain a *mobilityPattern* attribute. Sensors and actuators may be stationary or mobile and the mobility pattern may impinge on system observability and optimisation algorithm selection. The Sensor class contains one or more State classes and the Actuator class contains zero or more State and one or more Action classes. In many applications the actions currently selected are also state information.

Elements of this layer contain a *confidence* attribute that indicates the degree of confidence associated with individual sensor readings and is represented as a probability value between 0 and 1. This value represents a prior or unconditional probability associated with the values produced by the sensor equipment.

A domain modeller could specify an inductive loop sensor [6] to have a *confidence* attribute of 0.9 whereas a virtual loop using video imagery may have a reduced *confidence* of 0.75. These values reflect the developers confidence in the sensor infrastructure in the domain.

3.2.3 System States Layer

System States are composites of layer 1 and 2 elements. The set of system states are defined by the domain modeller and are usually derived from multisensor data.

The design of this layer is presented in Figure 4. Associated with each system-state element are *scope* and *observability* attributes. The *scope* attribute defines the physical region over the domain model to which the system state relates and is expressed as a spatial value that can have one of three values:

- Infrastructure Element Scope. A system state can be defined at the level of an individual element at the infrastructure layer with a location reference attribute that matches the associated infrastructure element location reference.
- Multiple Element Scope. A system state can be defined at the level of multiple infrastructure elements. The location reference attribute of such a system state is the minimum-bounding-rectangle of the multiple elements.
- Region Scope. A system state can be defined at the level of a region of the domain specified by the domain modeller and is assigned as a location reference a newly created geometry that matches the region.

[1] http://www.opengis.net/gml
[2] http://www.opengeospatial.org/docs/99-049.pdf
[3] http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

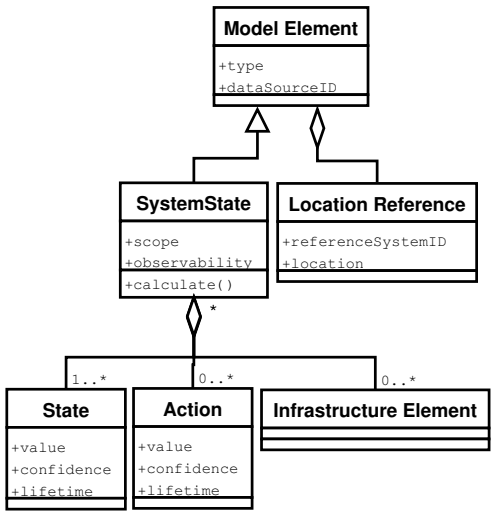

Figure 4. System States Layer

The *observability* of a system state is a measurement of the sensor spread and quality in the region of the domain model determined by the scope of the system state element. It is related to the confidence level associated with sensor readings relating to its constituent layer 2 elements.

The set of possible system states is constrained by the elements defined at layer 1 and 2 as it is only possible to define a system state over the existing elements of the underlying layers. By defining a system state, the modeller identifies the type of infrastructure and sensor and actuator layer data that is relevant to determining the value of the system state. The mapping from element type to the deployed sensor and actuator platform in the application environment is performed using spatial operations over the geometries associated with all layer elements.

This mapping or registration between sensor and actuator components and system states is used to support multisensor integration and fusion.

3.2.4 Domain-Specific Knowledge Layer

Layer 4 represents domain specific knowledge as prespecified action selection strategies that map relationships between model components that have been identified by the developer. This knowledge is encoded in the state-transition system.

An example of domain specific knowledge in the UTC domain is the traffic theory governing the relationship between optimum cycle time at a junction and flow rate at approaches to the junction [16].

In the absence of this domain knowledge or mapping we would have to use search and exploration techniques to learn this relationship. In a complex real world system the state-space size might be large enough to render such a task prohibitive, as evidenced in [8].

3.3 Observation Model

The observation model characterises the efficacy of the sensor and actuator infrastructure when determining system state and can be used as an input into multisensor integration and fusion techniques.

Multisensor integration and fusion involves high-level challenges such as the integration of multiple sensors at the system architecture and control level to more specific issues - possibly mathematical or statistical involved in the actual combination of multisensor information [10].

This challenge may be decomposed into three sub tasks: sensor selection, data fusion and estimation or inference about the system state. Layer 3 elements in the domain model contain a spatial mapping between a system state and its related set of sensor and actuator data. This mapping is used to support sensor selection. The *confidence* attribute associated with sensor and actuator data can act as an input into sensor fusion and state inference techniques.

The choice of fusion and inference algorithms used in Ubicomp application development is independent of the proposed domain and planning model.

A Ubicomp application constructed as a POMDP will use an observation model to construct a belief state which is a probability distribution over all possible system states. This belief space is then used when determining action selection [12]. Another approach is to use Dynamic Bayesian Networks for sensor fusion and state inference. Sensor data may be represented as evidence or observation nodes in a Bayesian network and the sensor *confidence* attribute can be used in determining a weighting for the associated evidence nodes [7]. In this scenario action selection is then based on an inferred system state rather than an observed belief state [13].

The use to which the information contained in the observation model is put, is related to the choice of optimisation algorithm for the application.

4 Domain Model Implementation

We have applied our domain modelling process to the UTC application domain and have constructed a model of the SCATS UTC system [14] that is deployed in Dublin City Centre. We have used the uDig geospatial platform [4] to develop a domain modelling tool. This tool provides the GIS support required to perform spatial operations that match sensor and actuator position and readings to system state scope. Layer 1 and 2 elements are encoded using the SFS

[4] http://udig.refractions.net

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

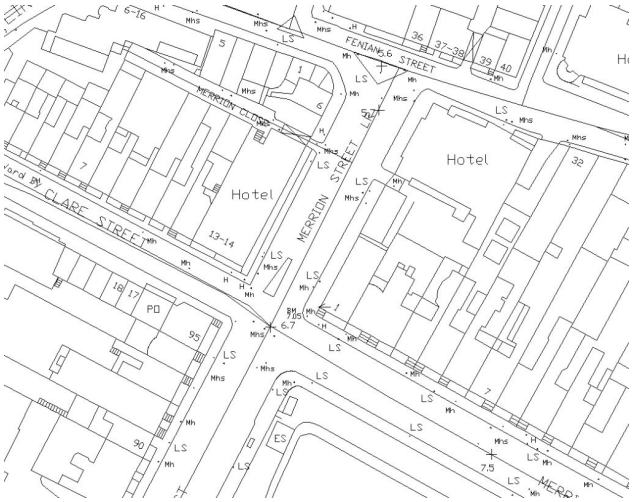

Figure 5. Excerpt from UTC Infrastructure Layer

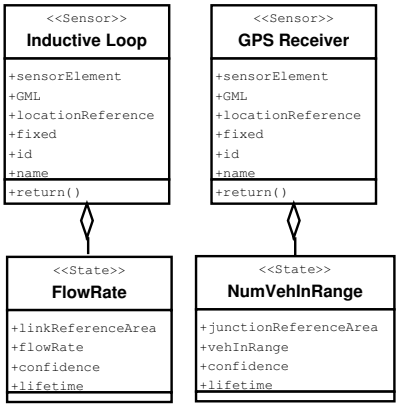

Figure 6. UTC Sensor Elements

data format and layer 3 and 4 elements are encoded using XML schema. All model elements are tagged with spatial attributes based on the Irish National Grid Coordinate reference system.

The Infrastructure Layer of our domain model contains Traffic Junction and Road Link elements that represent the road network in Dublin. Fig 5 shows a screen shot of Infrastructure Layer elements from a region of the road network adjacent to Trinity College in Dublin.

In addition to capturing the geometry and structure of the network, the Junction element specifies the set of traffic light phases or control sequences that can run at a junction and the Link element specifies the maximum rate at which vehicles can flow along the link. This value has been established from traffic theory to be 1800 vehicles per hour [16].

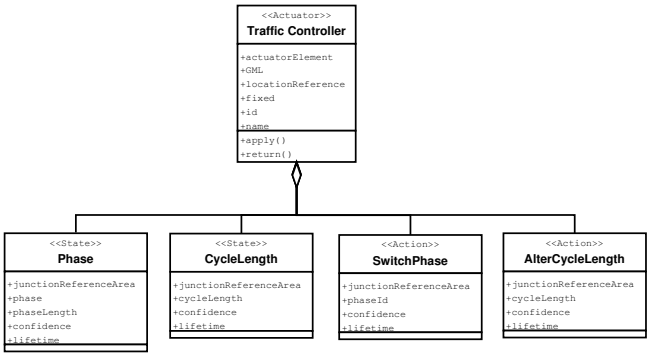

Figure 7. UTC Actuator Elements

Fig 6 shows two Sensor elements from Layer 2 of the model. The Inductive Loop and GPS Receiver sensors provide sensor data on the flow rate of a link and the number of vehicles in range of a junction respectively.

Fig 7 shows an Actuator element from Layer 2 of the model. The Traffic Controller actuator provides data on the current control phase and cycle-length at a traffic light junction. This actuator can act to switch phases and to alter the junction cycle length.

Fig 8 shows the *DegreeOfSaturation* system state from Layer 3 of the model. This system state is used by the optimisation strategy employed by SCATS application to control the traffic light settings in the UTC system [9]. It is a composite of Layer 1 and Layer 2 elements and is defined as:

$$x = q/Q$$

q is the actual flow rate in vehicles/hour; Q is the maximum achievable flow of an approach to a signal and is calculated from $Q = sg/C$; s is the saturation flow or the maximum rate at which traffic can flow along the link; g is the green time allocated to a traffic light control phase and C is the cycle time in operation at the traffic light junction.

This system state captures the ratio of effectively used green time to available green time at a junction. The SCATS policy is to keep this ratio at 0.9, i.e. traffic should flow through a junction for 90% of the portion of a green phase. The *DegreeOfSaturation (DOS)* on a link may be calculated as :

$$flowRate/\frac{saturationFlow \times phaseLength}{cycleLength}$$

This components of this system-state are:

- *flowRate* - a layer2 element
- *saturationFlow* - a layer 1 infrastructure element
- *phaseLength* - a layer 2 element

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

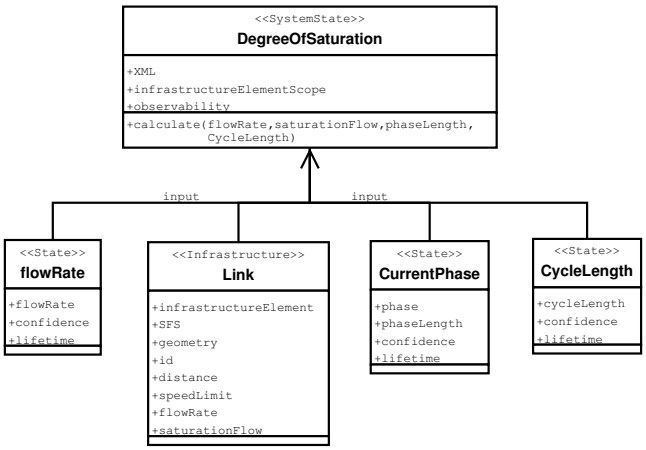

Figure 8. Sample UTC System States Layer

- *cycleLength* - a layer 2 element

SCATS uses the *DOS* system state as a metric for calibrating traffic light junction cycle times to maximise the throughput of traffic at a junction. A traffic junction has a number of approaches and SCATS calculates a *DOS* value for each approach to the junction.

In our domain model we assign each *DOS* system state an infrastructure element scope that matches the geometry of the road link element specified in layer 1. The overlapping spatial attributes allow us to identify the actual set of sensors and actuators deployed in the environment that are associated with each separate instance of the *DOS* system state on each approach to all junctions in the system. Once the sensors and actuators have been identified, the possibly varying degree of observability over a set of common system states can be calculated and used in sensor fusion and state inference techniques.

From traffic theory [16] the relationship between the optimum junction cycle time C and traffic flow Y is:

$$C = (1.5L + 5)/(1 - Y)$$

Y is the total flow ratio and is calculated as $\sum y$ where y is defined as the movement flow ratio $y = q/s$ where q is the demand or flow rate on an approach and s is the saturation flow or the maximum rate at which traffic can pass a point. L is the lost time for a signal intersection due to safety delays between phase switching.

By exploiting this domain specific knowledge the system can apply a control strategy to optimise traffic flow by altering junction cycle length times in response to variations in traffic flow.

4.1 Building a Planning Model

The domain model is parsed to extract the information required to construct a planning model. This process involves iterating over the set of system states to determine the infrastructure layer and sensor and actuator layer elements that are relevant to the application.

Algorithm 1 Parsing the Domain Model

Input: D {a Ubicomp domain model}

1: **for all** system states **do**
2: select layer1 and 2 elements within scope
3: calculate associated *confidence* attributes
4: seed state inference algorithms with *confidence* attributes
5: populate the sets S and A of states and actions
6: **end for**

Step 2 is implemented using the GIS platform. The registration of sensors and their associated confidence attributes facilitates multisensor integration and fusion algorithms.

In the UTC domain model the *flowRate* sensor reading is associated with the *(DOS)* system state. This reading is obtained from inductive loop sensors located along each road Link element. In this application there is a one to one mapping between the inductive loop sensor and the *flowRate* for each traffic light phase at all Junction elements. In this case there is no requirement for multisensor integration or fusion.

5 Conclusions

The proposed domain modelling technique has been developed to support Ubicomp development in environments characterised by partial observability. The domain model captures the information required to construct a planning model that acts as an input into the Ubicomp development process.

System states are defined as composites of the sensor, actuator and infrastructure layer elements. An application is defined across a subset of system states. All model elements are tagged with spatial attributes to automate the registration or association of deployed sensors and actuators to system states. This information coupled with the prior probabilities associated with sensor readings can support multisensor integration and fusion techniques that allow us to determine the degree of observability that exists over the set of system states.

To date we have constructed a domain model for a UTC application for Dublin city centre and have extracted a planning model that we have used in a road traffic optimisation application. The UTC domain model contains large data

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE

sets that are required to represent the infrastructure and sensor and actuator elements in the city region. However a lot of this data is available is standard geographic data formats from the City Council authorities. The multi-layer modelling approach and spatial operations provide a uniform and scalable method of evaluating the sensor coverage in Ubicomp domains.

We now wish to build domain models for a range of Ubicomp domains and explore how the planning model and observability metrics can be used as an aid in identifying appropriate optimisation algorithms for Ubicomp applications.

References

[1] B. Abdulhai, R. Pringle, and K. G. Reinforcement Learning for True Adaptive Traffic Signal Control. *Transportation Engineering.*, 129(3), May 2003.

[2] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, volume 2, pages 1023–1028, Seattle, Washington, USA, 1994. AAAI Press/MIT Press.

[3] J. Dowling, R. Cunningham, A. Harrington, E. Curran, and V. Cahill. Emergent consensus in decentralised systems using collaborative reinforcement learning. In O. Babaoglu, M. Jelasity, A. Montresor, C. Fetzer, S. Leonardi, A. van Moorsel, and M. van Steen, editors, *Post-Proceedings of the workshop SELF-STAR: Self-* Properties in Complex Information Systems, Hot Topics in Computer Science*, volume 3460 of *Lecture Notes in Computer Science*, pages 63–80. Springer-Verlag, May 2005.

[4] M. Ghallab, D. Nau, and P. Traverso. *Automated Planning Theory and Practice*. Morgan Kaufmann, 2004.

[5] L. Klein. *Sensor Technologies and Data Requirements for ITS*. Artech House, 2001.

[6] K. Korb and A. Nicholson. *"Bayesian Artificial Intelligence"*. Chapman and Hall, 2004.

[7] H. Lo and A. Chow. Control Strategies for Over-Saturated Traffic. *Transportation Engineering.*, 130(4), 2004.

[8] P. Lowrie. "The Sydney Co-ordinated Adaptive Traffic System - principles, methodology, algorithms". In *Proc. IEEE International Conference on Road Traffic Signalling*, pages 67–70, 1982.

[9] R. Luo and M. Kay. "Multisensor Integration and Fusion in Intelligent Systems". *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5), September/October 1989.

[10] M. Maurette. Mars rover autonomous navigation. *Auton. Robots*, 14(2-3):199–208, 2003.

[11] S. Russell and P. Norvig. *Artificial Intelligence - A Modern Approach*. Prentice Hall, 2 edition, 2003.

[12] A. Senart, M. Bouroche, G. Biegel, and V. Cahill. A component-based middleware architecture for sentient computing. In *ECOOP 2004 Workshop on Component-Oriented Approaches to Context-Aware Systems*, June 2004.

[13] A. Sims and K. Dobinson. "The Sydney Co-ordinated Adaptive Traffic System (scat) System Philosophy and Benefits". In *Proc. IEEE International Conference on Vehicular Technology*, volume VT-29, pages 130–137, 1980.

[14] J. Spall. *Introduction to Stochastic Search and Optimization*. Wiley InterScience, 2003.

[15] F. Webster. *Traffic Signal Settings - Technical Paper No 39*. Road Research Laboratory, London, UK, 1959.

[16] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, April 1997.

21st International Conference on
Advanced Information Networking and Applications Workshops (AINAW'07)
0-7695-2847-3/07 $20.00 © 2007 IEEE