Sensor Based Designs for Smart Space*

Jin Song Dong† Yuzhang Feng† Jing Sun‡ Jun Sun†

†Computer Science Department, National University of Singapore
{dongjs,fengyz,sunj}@comp.nus.edu.sg

‡Computer Science Department, The University of Auckland
j.sun@cs.auckland.ac.nz

Abstract

The challenge for specifying and designing complex sensor based smart space is how to precisely capture communicating behaviors, sensor constraints, and real-time system properties in a highly abstract, modular and integrated way. In particular, the high level design models for the smart systems need to capture not only concurrent interactions between various software control units and physical sensing devices, but also environmental and requirement constraints on sensors and sensor relations. In this paper, we demonstrate that a sensor based formalism can be succinctly applied to the design of smart space with multiple functionalities. Based on the formal model, essential properties within a particular domain or across different domains can also be verified.

1 Introduction

The recent development on sensor-based context-aware, ubiquitous and mobile computing has added new complexity and created new challenges in system design techniques for modelling and reasoning complex smart systems. The objectives of specifying and designing complex sensor based smart system is to precisely capture communicating behaviors, sensor constraints, and real-time system properties in a highly abstract, modular and integrated model on which important properties can be verified. In particular, the high level design models for the smart systems need to capture not only concurrent interactions between various software control units

*This work is supported in part by the InfoComm and InfoTech Initiative research grant "Reliable Software Design and Development for Sensor Network Systems"

and physical sensing devices, but also environmental and requirement constraints on sensors and sensor relations. For example, a smart meeting room may have a number of autonomous control units that control lighting/security/aircondition devices, and even mobile devices (mobile phones attached with a RFID tag may automatically turn into the silence mode when they enter the meeting room). This kind of application consists of a number of different sensors. The above example may consist of a motion sensor for detecting any occupants, a light sensor for detecting the current outdoor lighting intensity and RFID tracking sensor to identify all mobile devices/users attached with a RFID tag. It is important for a design model to precisely capture the requirement constraints on sensors, e.g. room sensor value for outdoor light intensity will never reach 100 percent illumination requirement for the room, and sensor relations, e.g. whenever a RFID tag (attached to mobile device or user) is tracked by the tracking unit, the motion sensor must detect occupants.

To facilitate design analysis and review process effectively, the design model should focus on appropriate abstraction level. Certain low level implementation details can be abstracted away. For example, in our experience [11] the communications between mobile device and RFID tracking unit may involve a Java program in the mobile device to send an active query to a server (and then wait for a notification if the location is changed) through BlueTooth technology. However, those implementation details are best hidden from the abstract design so that more important system properties can be clearly reasoned.

In this paper, we demonstrate that a sensor based formalism can be succinctly applied to the design of smart space with multiple functionalities. Based on the formal model, essential properties within a particular domain or across different domains can also be verified.

2 TCOZ

Timed Communicating Object Z (TCOZ) [9, 7] is essentially a blending of Object-Z [3] with Timed CSP [1], for the most part preserving them as proper sub-languages of the blended notation. The essence of this blending is the identification of Object-Z operation specification schemas with terminating CSP processes. Thus operation schemas and CSP processes occupy the same syntactic and semantic category, operation schema expressions may appear wherever processes may appear in CSP and CSP process definitions may appear wherever operation definitions may appear in Object-Z. The primary specification structuring device in TCOZ is the Object-Z class mechanism.

In this section we briefly consider various aspects of TCOZ. A detailed introduction to TCOZ and its Timed CSP and Object-Z features may be found elsewhere [9]. The formal semantics of TCOZ is also documented [8, 10].

2.1 A model of time

In TCOZ, all timing information is represented as real valued measurements in *seconds*, the SI standard unit of time. We believe that a mature approach to measurement and measurement standards is essential to the application of formal techniques to systems engineering problems. In order to support the use of standard units of measurement, extensions to the Z typing system suggested by Hayes and Mahony [4] are adopted. Under this convention, time quantities are represented by the type

$$\mathbb{T} == \mathbb{R} \odot \mathsf{T},$$

where $\mathbb{R}$ represents the real numbers and T is the SI symbol for dimensions of time. Time literals consist of a real number literal annotated with a symbol representing a unit of time. All the arithmetic operators are extended in the obvious way to allow calculations involving units of measurement.

2.2 Interface – channels, sensors and actuators

CSP channels are given an independent, first class role in TCOZ. In order to support the role of CSP channels, the state schema convention is extended to allow the declaration of communication channels. If c is to be used as a communication channel by any of the operations of a class, then it must be declared of type **chan** in the state schema. Channels are type heterogeneous and may carry communications of any type. Contrary to the conventions adopted for internal state attributes, channels are viewed as shared (global) rather than as encapsulated entities. This is an essential consequence of their role as communications interfaces *between* objects. The introduction of channels to TCOZ reduces the need to reference other classes in class definitions, thereby enhancing the modularity of system specifications.

Complementary to the synchronising CSP channel mechanism and inspired by control thoery, TCOZ also adopts a non-synchronising shared variable mechanism [6]. A declaration of the form $s : X$ **sensor** provides a channel-like interface for using the shared variable s as an input. A declaration of the form $s : X$ **actuator** provides a local-variable-like interface for using the shared variable s as an output. Sensors and actuators may appear either at the system boundary (usually describing how

global analog quantities are sampled from, or generated by the digital subsystem) or else within the system (providing a convenient mechanism for describing local communications which do not require synchronisations). The shift from closed to open systems necessitates close attention to issues of control, an area where both Z and CSP are weak [12]. We believe that TCOZ with the **sensor** (and **actuator**) can be a good design language for smart sensor based hybrid systems.

2.3 Active objects

Active objects have their own thread of control, while passive objects are controlled by other objects in a system. In TCOZ, an identifier MAIN (non-terminating process) is used to determine the behaviour of active objects of a given class [2]. The MAIN operation is optional in a class definition. It only appears in a class definition when the objects of that class are active objects. Classes for defining passive objects will not have the MAIN definition, but may contain CSP process constructors. If ob_1 and ob_2 are active objects of the class C, then the independent parallel composition behaviour of the two objects can be represented as $ob_1 \mid\mid\mid ob_2$, which means $ob_1.\text{MAIN} \mid\mid\mid ob_2.\text{MAIN}$

2.4 Semantics of TCOZ

A separate paper details the blended state/event process model which forms the basis for the TCOZ semantics [8, 10]. In brief, the semantic approach is to identify the notions of operation and process by providing a process interpretation of the Z operation schema construct. TCOZ differs from many other approaches to blending Object-Z with a process algebra in that it does not identify operations with events. Instead an unspecified, fine-grained collection of state-update events is hypothesized. Operation schemas are modelled by the collection of those sequences of update events that achieve the state change described by the schema. This means that there is no semantic difference between a Z operation schema and a CSP process. It therefore makes sense to also identify their syntactic classes.

The process model used by TCOZ consists of sets of tuples consisting of: an *initial* state; a *trace* (a sequence of time stamped events, including update-events), a *refusal* (a record of what and when events are refused by the process), and a *divergence* (a record of if and when the process diverged). The trace/refusal pair is called a *failure* and the overall model the state/failures/divergences model. The state of the process at any given time is the initial state updated by all of the updates that have occurred up to that time. If an event trace terminates (that is if a ✓ event occurs), then the state at the time of termination is called the *final* state.

The process model of an operation schema consists of all initial states and update traces (terminated with a ✓) such that the initial state and the final state satisfy the relation described by the schema. If no legal final state exists for a given initial state, the operation diverges immediately. An advantage of this semantics is that it allows CSP process refinement to agree with Z operation refinement.

2.5 Network topologies

The syntactic structure of the CSP synchronisation operator is convenient only in the case of pipe-line like communication topologies. Expressing more complex communication topologies generally results in unacceptably complicated expressions. In TCOZ, a graph-based approach is adopted to represent the network topology [5]. For example, consider that processes A and B communicate privately through the interface ab, processes A and C communicate privately through the interface ac, and processes B and C communicate privately through the interface bc. This network topology of A, B and C may be described by

$$\Big|\Big|(A \xleftrightarrow{ab} B;\ B \xleftrightarrow{bc} C;\ C \xleftrightarrow{ca} A).$$

Other forms of lax usage allow network connections with multiple channels above the arrow, for example if processes D and F communicate privately through the channel/sensor-actuator df_1 and df_2, then

$$\Big|\Big|(D \xleftrightarrow{df_1, df_2} F).$$

3 Smart Space: intelligent control over lighting and mobile phones

The aims of our smart space design are to save electric energy and provide autonomous control over the receiving modes of mobile phones. In most meeting rooms (or libaray/museum) today, lights are controlled manually. Electrical energy is wasted by lighting rooms that are not occupied and by not adjusting light levels relative to the daylight. The first component of our smart room is an intelligent light control unit. It can detect the occupation of the building through a motion sensor, turn on or turn off the lights automatically. It is able to tune illumination in the building according to the outside light level. When users leave a room (leaving it empty) and the detector senses no movement, the light control unit waits a pre-defined absence time and then turns off the light group. The second concern is related to the widespread use of mobile phones which makes voice

communication available anytime, anywhere, but also raises many social problems such as phones ringing during meetings. Normally users often have to configure the settings of mobile phones according to their circumstances to avoid inappropriate usage. However, manual configuration causes frequent interactions with mobile phone, imposing significant user distractions. To address this issue, the second component of our smart system is designed to automatically detecting mobile phone/user's location (via RFID tracking system). For example, if a mobile user (wearing a RFID tag) is determined to be in the meeting room then a control message will be sent from the mobile tracking system to the mobile phone so that a silent mode will be automatically selected. The silence mode can be either Vibration or Voice-mail mode depending on individual user's profile which can be updated dynamically.

4 Sensor Based Modelling

4.1 Sensors, Sensor Constraints and Sensing Patterns

The standard TCOZ communications interface is the CSP channel, which represents a sequence of discrete synchronization between system and environment. One common model for hybrid smart system interfaces is the continuous, differentiable function. We adopt an approach by providing standardized mechanisms in TCOZ for converting from the discrete to the continuous and vice versa, thus granting TCOZ process classes to present a continuous-function interface to their environments. This allows subsystems specified in TCOZ to fit seamlessly into the overall design of hybrid smart systems.
For example, the motion detector for a room can be specified by a declaration of the form

$$motion : OccupyState \,\textbf{sensor}, \text{ where } OccupyState ::= Occupied \mid Empty$$

which declares $motion$ to be a continuous-function interface with public type $\mathbb{R}\,\mathsf{s} \to OccupyState$.
Internally, $motion$ takes the syntactic role of a CSP channel. The relationship between the public continuous-function variable and the internal channel is that whenever a value v is communicated on the internal channel at a time t, that value must be equal to the value of the continuous function at that time, that is $motion(t) = v$.
The light group intensity can be specified by a declaration of the form

$$i : Illumination \,\textbf{actuator}, \text{ where } Illumination ::= 1\,..\,100$$

This declaration also introduces i as a public continuous-function variable, but in this case the internal role is that of the local state variable. Thus i may be updated and appear in the delta list of operations and any other place where a local variable may appear.

Sensor Constraints and Sensing Patterns

Various sensor constraints can be generally captured in the form of

$$\forall\, t : \mathbb{T} \bullet C(sensor_a(t))$$

where C can be an invariant constraint on sensor (or well tested environmental assumption). Inter-sensor relationships can also be captured in a similar form of

$$\forall\, t : \mathbb{T} \bullet R(sensor_a(t), sensor_b(t))$$

where R can be a relational constraint for $sensor_a$ and $sensor_b$ in predicate form. Smart process involving sensors can have various behavior patterns. For example,

- *Periodic Sensing* pattern can be specified by the following frame:

  $$\mu\, P \bullet [v : \mathbb{R}] sensor_a?v \rightarrow (P_1 \,\text{DEADLINE}\, 1\,\mathsf{s} \bullet \text{WAITUNTIL}\, 1\,\mathsf{min});\ P$$

  where the $sensor_a$ will sense any real number value in one minute interval (provided the sub process P_1 must terminate within one second. Patterns for sensors to periodically wake up can also be easily specified by this kind of frame.

- *Conditional sensing* pattern can be specified by the following frame:

  $$...[v : \mathbb{R} \mid c(v)] \bullet sensor_b?v \rightarrow P_2$$

  where $c(v)$ is a condition on v (e.g. $5 < v < 10$) that $sensor_b$ wants to detect.

There are obviously other sensing patterns, e.g. timeout sensing and interrupt sensing. We will demonstrate the various sensing design patterns in the following smart space modelling case study.

4.2 Model of the Smart Space

LightGroup

i : *Illumination* **actuator**
$dimmer$: **chan**

MAIN $\widehat{=}$ $\mu\, N \bullet [n : Illumination] \bullet dimmer?n \rightarrow i := n;\ N$

Class $LightGroup$ defines the data state and behaviors of the *lights*. The light level is represented as a state variable $i : Illumination$. $dimmer$ is defined as a channel connecting the light group to the light control unit. A MAIN process indicates that $LightGroup$ defines an active object, which has its own thread of control. Whenever a message is received on the channel $dimmer$, the light level is updated with the new light level. The light level update can occur repeatedly, due to the recursion.

LightControlUnit

$dimmer$: **chan**
$motion$: $OccupyState$ **sensor**
$odsensor$: *Illumination* **sensor**
$absenT$: $\mathbb{T}$

$\forall\, t : \mathbb{T} \bullet odensor(t) < 100$

Adjust

$n?$: *Illumination*
$dim!$: *Illumination*

$dim! + n? = 100$

$Ready \mathrel{\widehat{=}} motion?Occupied \rightarrow On$
$Regular \mathrel{\widehat{=}} \mu\, R \bullet [n : Illumination] \bullet odsensor?n \rightarrow$
$\quad (Adjust \bullet \text{DEADLINE}\ 0.5\,\text{s} \bullet \text{WAITUNTIL}\ 1\,\text{s});\ dimmer!dim \rightarrow R$
$On \mathrel{\widehat{=}} Regular \bigtriangledown motion?Empty \rightarrow OnAgain$
$OnAgain \mathrel{\widehat{=}} (motion?Occupied \rightarrow On) \triangleright \{absenT\}\ Off$
$Off \mathrel{\widehat{=}} dimmer!0 \rightarrow Ready$
MAIN $\widehat{=}$ *Off*

The light control unit communicates with the light group by declaring channels with the same name, monitors the signals from its sensors and sends proper signals to the light group. The sensor $motion$ detects movement in the building. The

sensor $odsensor$ monitors the light level outside. The class invariant

$$\forall\, t : \mathbb{T} \bullet odensor(t) < 100$$

is a sensor constraint which captures an environmental assumption (i.e. due to the building window size, outdoor light intensity can never reach 100 percent room Illumination requirement). The process $Regular$ is periodically (one second) sensing out-door lighting and outputting a proper light level to the light group.
At start-up, the light control unit sets the light level to 0 by sending value 0 through channel $dimmer$. Once some movement is detected by the motion sensor ($motion?Occupied$), the light control unit starts to repeatedly read the outside light level ($odsensor?n$) and then adjust the light level accordingly. If the motion sensor detects no movement ($motion?Empty$), the light control unit waits a predefined absence time ($absenT$, captured by the timeout operator $\triangleright$) and then turns off the light. However, if some movement is detected before the absence time, the control unit continues adjusting the light level.

MobileTrackingUnit

$tracking : \mathbb{P}\, RFID\_tag$ **sensor**
$recorded : \mathbb{P}\, RFID\_tag$
$signal$: **chan**

INIT
$recorded = \varnothing$

Update
$\Delta(recorded)$
$ms? : \mathbb{P}\, RFID\_tag$

$recorded' = ms?$

$$\text{MAIN} \mathrel{\widehat=} \mu\, MU \bullet [ms : \mathbb{P}\, RFID\_tag \mid ms \neq recorded] \bullet tracking?ms \rightarrow (\Big|\Big|_{m:(ms-recorded)} signal!(m, Silence) \rightarrow \text{SKIP} \parallel \Big|\Big|_{m:(recorded-ms)} signal!(m, Back) \rightarrow \text{SKIP});\ Update;\ MU$$

The mobile tracking unit monitors all the RFID tags within range through the sensor $tracking$. It records the set of RFID tags that have been previously detected and communicates with individual mobile phones through channel $signal$. Initially, no RFID tag is recorded. The operation $Update$ replaces the recorded RFID tags by the newly detected ones. Whenever a new set of RFID tags has been detected (conditional sensing pattern), for every RFID tag that has not been detected previously,

a control message is sent to the corresponding mobile phone so that a silent mode will be automatically selected. For every RFID tag that the tracking sensor has lost track of, a control message is sent to the corresponding mobile phone so that the mobile phone is set to its previous mode.

$$MobilePMode \mathrel{\widehat=} Vibration \mid Ring \mid RingAndVibration \mid VoiceMail$$

MobileP

$id : RFID\_tag$
$signal$: **chan**
$mode, pre\_mode, silence\_mode : MobilePMode$

ManualUpdate

$\Delta(mode)$
$m? : MobilePMode$

$mode' = m?$

ChangeSilenceProfile

$\Delta(silence\_mode)$
$m? : Vibration \mid VoiceMail$

$silence\_mode' = m?$

AutoSilence

$\Delta(mode, pre\_mode)$

$pre\_mode' = mode$
$mode' = silence\_model$

SwitchBack

$\Delta(mode)$

$mode' = pre\_mode$

$\text{MAIN} \mathrel{\widehat=} \mu\, H \bullet (ManualUpdate \,\Box\, ChangeSilenceProfile \,\Box$
$signal?(id, Silence) \rightarrow AutoSilence;\ signal?(id, Back) \rightarrow SwitchBack);\ H$

$MobilePMode$ defines the different modes of mobile phones. A mobile phone is associated with the RFID tag. It also records its current mode ($mode$), its previous mode (pre_mode) and its silence profile ($silence\_mode$, either $Vibration$ or $VoiceMail$). The mode of the mobile phone can be manually updated by the operation $ManualUpdate$. The silence profile can be updated by the operation $ChangeSilenceProfile$. Once a signal is received through channel $signal$ from the mobile tracking system with the right RFID tag, if the message content is $Silence$, the mobile phone is set to silence mode and its previous mode is recorded in the variable pre_mode. If the message content is $Back$, the mobile phone is set back to its previous mode.

MobilePs

$phones : \mathbb{P}\, MobileP$
$ids : \mathbb{P}\, RFID\_tag$

$ids = \{p : phones \mid p.id\}$
$\#phones = \#ids$
[unique association of RFID tag with phones]

MAIN $\widehat{=}$ $|||\ p : phones$

Class $MobilePs$ defines a group of mobile phones. Its behavior is the interleaving of all the mobile phones.

SmartMeetingSpace

$lg : LightGroup$
$lcu : LightControlUnit$
$mps : MobilePs$
$mtu : MobileTrackingUnit$

$mtu.recorded \subseteq mps.ids$
$\forall\, t : \mathbb{T} \bullet mtu.tracking(t) \neq \varnothing \Rightarrow lcu.motion(t) = Occupied$

MAIN $\widehat{=}$ $\Big|\Big|(mps \overset{signal}{\longleftrightarrow} mtu;\ lcu \overset{dimmer}{\longleftrightarrow} lg)$

The smart meeting space consists of a light group (lg), a light control unit (lcu), a set of mobile phones (mps) and a mobile tracking unit (mtu). The system is constrained such that the mobile tracking unit tracks only those RFID tags associated with the mobile phones. The class invariant

$$\forall\, t : \mathbb{T} \bullet mtu.tracking(t) \neq \varnothing \Rightarrow lcu.motion(t) = Occupied$$

captures a relationship between two different sensors, i.e., whenever a mobile user is tracked by the tracking unit, the motion sensor must detect some movement.

5 Reasoning about the smart space properties

One of the promising advantages of using a formal modelling approach is that verifications of the specified system properties by means of sound proofs can be

made possible. We demonstrate the reasoning of two key properties, one within the mobile domain and one across the mobile and the light domain.

1. Intra-domain property
   All the mobile phones in a smart meeting space should be in silence mode, which ensures that there should be no unexpect interruptions during the meeting time. This property can be formally expressed from the model as:

$$SmartMeetingSpace :: \forall\, m : mps.phones \bullet \quad m.id \in mtu.recorded \Rightarrow m.mode = m.silence\_mode \qquad \text{[P1]}$$

2. Inter-domain property
   When there are mobile users in a smart meeting space, its lights should be on. This property can be formally expressed from the model as:

$$SmartMeetingSpace :: mtu.recorded \neq \varnothing \Rightarrow lg.i > 0 \qquad \text{[P2]}$$

In order to prove *P1*, we first need to show that *once a mobile phone is switched into a silent model it will remain in its status until a switch back signal is received*. This can be formally stated as follows.

$$\begin{aligned} &SmartMeetingSpace :: \forall\, m : mps.phones \bullet \qquad \text{[P1.1]} \\ &\quad (m.id \in mtu.recorded \\ &\quad \wedge\ mtu.(signal!(m.id, Silence)) \in (RFID\_tag \times \{Silence\}) \\ &\quad \wedge\ mtu.(signal!(m.id, Back)) \notin (RFID\_tag \times \{Back\})) \\ &\quad \Rightarrow m.mode = m.silence\_mode \end{aligned}$$

Proofs of the property *P1.1* can be constructed as follows.

$$\frac{\begin{array}{l} MobileTrackingUnit :: STATE \vdash signal \in \textbf{chan} \wedge \textsc{Main} \vdash \\ \quad (signal!(id, Silence) \in (RFID\_tag \times \{Silence\}) \wedge \\ \quad signal!(id, Back) \notin (RFID\_tag \times \{Back\})) \\ MobileP :: STATE \vdash signal \in \textbf{chan} \\ SmartMeetingSpace :: STATE \vdash \\ \qquad (mtu \in MobileTrackingUnit \wedge mps \in MobilePs) \\ \quad \wedge \textsc{Main} \vdash mps \overset{signal}{\longleftrightarrow} mtu \\ MobilePs :: STATE \vdash phones \in \mathbb{P}\, MobileP \\ \quad \wedge \textsc{Main} \vdash (p \in phones \wedge ||| \; p) \end{array}}{\begin{array}{l} MobileP :: \textsc{Main} \vdash (signal?(id, Silence) \in (RFID\_tag \times \{Silence\}) \\ \quad \wedge signal?(id, Back) \notin (RFID\_tag \times \{Back\})) \\ MobileP :: MAIN \vdash \\ \quad (signal?(id, Silence) \in (RFID\_tag \times \{Silence\}) \wedge \\ \quad signal?(id, Back) \notin (RFID\_tag \times \{Back\})) \\ \qquad \Rightarrow (mode' = silence\_mode \wedge pre\_model' = mode) \end{array}} [\ Channel\ ]$$

$$\frac{}{MobileP :: MAIN \vdash mode' = silence\_mode}$$

Therefore, the property *P1* can be proved using induction on the behaviors of the active object *MobileTrackingUnit* as follows.

Proof: Initially, the recorded *RFID* set is empty, i.e.,

$$MobileTrackingUnit :: INIT \vdash recorded = \varnothing \wedge \textsc{Main} \vdash (ms \neq \varnothing \Rightarrow \forall\, m \in ms \bullet signal!(m, Silence) \in (RFID\_tag \times \{Silence\})$$

If a new set of *RFID* has been detected by the sensor, i.e., '$ms - \varnothing \neq \varnothing$', silence mode requests through the channel ***signal!.(m, Silent)*** will be sent to each individual mobile devices in the new record set *ms*. From the lemma ***P1.1*** we know that this will further trigger the *AutoSilence* operation in each mobile device, which set the post-state of recorded mobile phones to silent mode. Thus property *P1* holds for the base case.

Assuming the property *P1* holds at any particular sequence of the execution in the *MobileTrackingUnit* object. Let us consider its next possible behavior i.e. if a different set of *RFID* is detected by the sensor.

$$\frac{\begin{array}{l} MobileTrackingUnit :: \textsc{Main} \vdash ms \neq recorded \\ \Rightarrow (\forall\, m \in (ms - recorded) \bullet \\ \qquad signal!(m, Silence) \in (RFID\_tag \times \{Silence\} \wedge \\ \quad \forall\, n \in (recorded - ms) \bullet \\ \qquad signal!(n, Back) \in (RFID\_tag \times \{Back\} \wedge \\ \quad (recorded - ms) \cap ms = \varnothing) \end{array}}{\begin{array}{l} SmartMeetingSpace :: \forall\, m : mps.phones \bullet \\ \quad m.id \in mtu.recorded \Rightarrow m.mode = m.silence\_mode \end{array}}\ [\ P1.1\ ]$$

Note that at any point, the set of detected mobile phones and the ones that left the meeting place are disjoint, i.e., *'(recorded-ms)∩ ms=∅'*. Thus the original set of mobile phones that is still in the meeting space will remain in their silent mode status, since no switch back signals could be sent. Hence, we conclude that the property *P1* holds for the next continuous behavior of the mobile tracking unit. According to the induction rule, *P1* is true for all cases in the model.
Similarly, the proof of the property *P2* can be constructed as follows.

$$\frac{\begin{array}{l} SmartMeetingSpace :: STATE \vdash (lg \in LightGroup \wedge \\ \quad lcu \in LightControlUnit \wedge mtu \in MobileTrackingUnit \wedge \\ \quad \forall\, t : \mathbb{T} \bullet mtu.tracking(t) \neq \varnothing \Rightarrow lcu.motion(t) = Occupied) \\ \quad \wedge\ \textsc{Main} \vdash lcu \xleftarrow{dimmer} l \end{array}}{\begin{array}{l} LigtControlUnit :: STATE \vdash (dimmer \in \textbf{chan} \wedge \\ \qquad \forall\, t : \mathbb{T} \bullet odensor(t) < 100) \\ \quad Ready \vdash (motion?Occupied \in (Occupied \mid Empty) \wedge \\ \qquad motion?Empty \notin (Occupied \mid Empty)) \\ \qquad \Rightarrow odsensor?n \in Illumination \wedge \\ \quad Regular \vdash odsensor?n \in Illumination \Rightarrow dim + n = 100 \wedge \\ \qquad dimmer!dim \in Illumination \\ LightGroup :: STATE \vdash dimmer \in \textbf{chan} \end{array}}\ [\ Invariant\ ]$$

$$\frac{\begin{array}{l} LightGroup :: \textsc{Main} \vdash dimmer?dim \in Illumination \wedge dim > 0 \\ LightGroup :: \textsc{Main} \vdash dimmer?n \in Illumination \Rightarrow i := n \end{array}}{LightGroup :: \textsc{Main} \vdash i > 0}\ [\ Channel\ ]$$

Because the RFID tracking sensor and the motion detecting sensor are tightly coupled with each other, whenever a *RFID* is detected in the meeting space the motion sensor in the room will be set to the '*Occupied*' status, which will further cause the lights in the room to remain on.

6 Summary

In this paper, we have demonstrated that an integrated formalism, TCOZ with sensor and sensing pattern frameworks can be effectively applied to the design of smart space systems. The communicating behaviors, sensor constraints and real-time system properties can be captured in a highly abstract and modular style. Based on the formal model, essential properties within a particular domain or crossing different domains can also be verified. Future work may bring us to investigate complex sensor network systems, i.e., reconfiguration ability may be handled by introducing hybrid channels and hybrid sensors/actuators that can carry not only data but also process/programs.

Acknowledgements

We would like to thank Brendan Mahony, Xiaohang Wang and Daqing Zhang for the past joint research. This work is supported in part by the InfoComm and InfoTech Initiative research grant "Reliable Software Design and Development for Sensor Network Systems".

References

[1] J. Davies and S. Schneider. A brief history of Timed CSP. *Theoretical Computer Science*, 138, 1995.

[2] J. S. Dong and B. Mahony. Active Objects in TCOZ. In J. Staples, M. Hinchey, and S. Liu, editors, *the 2nd IEEE International Conference on Formal Engineering Methods (ICFEM'98)*, pages 16–25. IEEE Press, December 1998.

[3] R. Duke, G. Rose, and G. Smith. Object-Z: a Specification Language Advocated for the Description of Standards. *Computer Standards and Interfaces*, 17:511–533, 1995.

[4] I. J. Hayes and B. P. Mahony. Using units of measurement in formal specifications. *Formal Aspects of Computing*, 7(3), 1995.

[5] B. Mahony and J. S. Dong. Network Topology and a Case Study in TCOZ. In J. Bowen, A. Fett, and M. Hinchey, editors, *The 11th International Conference of Z Users*, volume 1493 of *Lecture Notes in Computer Science*, pages 308–327, Berlin, Germany, September 1998. Springer-Verlag.

[6] B. Mahony and J. S. Dong. Sensors and Actuators in TCOZ. In J. Wing, J. Woodcock, and J. Davies, editors, *FM'99: World Congress on Formal Methods*, Lect. Notes in Comput. Sci., pages 1166–1185, Toulouse, France, September 1999. Springer-Verlag.

[7] B. Mahony and J. S. Dong. Timed Communicating Object Z. *IEEE Transactions on Software Engineering*, 26(2):150–177, February 2000.

[8] B. Mahony and J. S. Dong. Deep Semantic Links of TCSP and Object-Z: TCOZ Approach. *Formal Aspects of Computing*, 13(2):142–160, 2002.

[9] B. P. Mahony and J. S. Dong. Blending Object-Z and Timed CSP: An introduction to TCOZ. In K. Futatsugi, R. Kemmerer, and K. Torii, editors, *The 20th International Conference on Software Engineering (ICSE'98)*, pages 95–104, Kyoto, Japan, April 1998. IEEE Press.

[10] S. C. Qin, J. S. Dong, and W. N. Chin. A Semantic Foundation of TCOZ in Unifying Theory of Programming. In *Proceedings of 12th Internation Symposium on Formal Methods E urope: FM'03*, Pisa, Italy, September 2003. LNCS, Springer-Verlag.

[11] X. Wang, D. Zhang, J. S. Dong, C. Chin, and S. Hettiarachchi. Semantic Space: An Infrastructure for Smart Spaces. *IEEE Pervasive Computing*, 3(3):32–39, July 2004.

[12] P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM Trans. Software Engineering and Methodology*, 6(1):1–30, January 1997.

A TCOZ glossary

Notation	Explanation
c : **chan**	declare c to be a channel
a : **actuator**	declare a to be a actuator
s : **sensor**	declare s to be a sensor
$\bot$	divergent process
STOP	deadlocked process
SKIP	terminate immediately

continued on next page

Notation	Explanation
WAIT t	delay termination by t
$a \rightarrow P$	communicate a then do P
$a@t \rightarrow P$	communicate a at time t then do P
$[t : \mathbb{T}] \bullet a@t \rightarrow P$	record time of a event in variable t
$c.a$	communicate a on channel c
$c?a$	input a on channel c
$c!a$	output a from channel c
$[b] \bullet P$	enable P only if b
$P;\ Q$	perform P until termination, then perform Q
$P \,\Box\, Q$	perform the first enabled of P and Q
$[i : I] \bullet P$	perform P with first enabled value of i (indexed external choice)
$P \sqcap Q$	perform either of P and Q
$[i! : I];\ P$	perform P with any value of i (indexed internal choice)
$v := e$	syntactic sugar for $[\Delta v \mid v' = e]$
$P \setminus A$	hide the events A from the environment of P
$P \,\|[\, A \,]\|\, Q$	synchronise P and Q on events from A
$(\Big\|\, p_1, \ldots, p_n \bullet \ldots;\ p_i \overset{A}{\longleftrightarrow} p_j;\ \ldots)$	network topology abstraction with parameters $p_1, \ldots, p_n$ and network connections including p_i communicating with p_j on private channels from A
$P \,\|\|\|\, Q$	P and Q running without sychronisations

continued on next page

Notation	Explanation
$P \triangleright \{t\}\ Q$	if P does not begin by time t, perform Q instead
$P \swarrow \{t\}\ Q$	perform P until time t, then transfer control to Q
$P \bigtriangledown e \rightarrow Q$	perform P until exception e, then transfer control to Q
$P \bullet \textsc{Deadline}\ t$	behaviours of P which terminate before time t
$P \bullet \textsc{WaitUntil}\ t$	after P idle until time t