A Semantic Framework for Reconfiguration of Instrumented Cyber Physical Spaces

Minyoung Kim[1], Daniel Massaguer[1], Nikil Dutt[1], Sharad Mehrotra[1], Shangping Ren[2],
Mark-Oliver Stehr[3], Carolyn Talcott[3], Nalini Venkatasubramanian[1]
[1]Univ. of California, Irvine {minyounk, dmassagu, dutt, sharad, nalini}@uci.edu
[2]Illinois Institue of Tech. ren@iit.edu
[3]SRI International {stehr, clt}@sri.com

1 Vision and Motivation - Future Cyber Physical Spaces

With recent advances in embedded computing, networking, and related information technologies, it is now feasible to deploy a variety of sensing devices, communication networks and IT services in the real world, creating instrumented smart spaces. Such instrumented physical spaces may contain variety of sensors including optical sensors, acoustic sensors, accelerometers, GPS devices, cell phones, RFID, Motes, as well as specialized sensors such as people-counters and load-cells that enable us to monitor the state of the physical world and its artifacts as well as activities that ensue in such a world. The multimodal data capture devices in these physical spaces are interconnected by a variety of standard (and possibly proprietary) communication networks, including wired Ethernet, cellular, WiFi(802.11b), Bluetooth, Zigbee, WiMAX, MANETS, mesh networks and power-line networks. The ability to integrate sensing and communication platforms with large scale distributed storage/compute facilities and software services enables the creation of **Instrumented Cyber Physical Spaces** (ICPSs) that are technologically cutting-edge and exemplary of highly-instrumented spaces of the future.

A variety of applications and services execute on such ICPS, often simultaneously; examples include asset monitoring, surveillance, social networking, facilities management and emergency response. Services such as medical triaging services, emergency response services, etc. involve human processes and decision making where humans in the loop determine how the application evolves based their awareness of the situation and infrastructure. In these applications, users in different roles execute on varying end-user devices (e.g., desktops, PDAs, cellphones, large scale display) and present information needs that vary in type, quality, and modality requirements. Applications dictate application specific constraints on the timeliness and accuracy/quality at which information must be captured and delivered from the infrastructure. Technologies and software systems deployed currently to manage such instrumented spaces are often designed and engineered for specific applications. Sensor programming frameworks often require knowledge of the low level sensing infrastructure (types and attributes of sensors). Repurposing the infrastructure, its software and hardware resources dynamically to realize different application functionalities is difficult.

In this context, we are developing frameworks that can customize the operations of ICPS to meet the varying needs of applications and users. Our approach to developing flexible and efficient cyber physical spaces is based on the principles of computational reflection, including an observe-analyze-adapt philosophy. The novelty of our approach lies in developing (a) real world systems that themselves follow and observe-analyze-adapt paradigm at an operational level, and (b) real world applications that are able to capture application concepts and execution at multiple levels using events as a fundamental abstraction and event-based semantics as a unifying framework.

2 Approach

The design of flexible software systems for ICPS poses significant challenges that must be overcome. Firstly, the sheer scale and heterogeneity of devices and components in ICPS that must be monitored, maintained and adapted for efficient operation of the space and its applications introduces complexities. Understanding the characteristics, constraints and capabilities of multiple sensing devices and networks is

key to determining what portion of the infrastructure to use for a specific task, when to use it and control parameters that will dictate device operation is a daunting task. Second, building physical systems that must operate under ever-changing conditions requires a shift in design methodology that will enable the design, development and deployment of physical systems with adaptivity as a basic design criteria. Thirdly, sensing and translating sensor outputs into observation about real world is fraught with errors and uncertainties.

We outline the requirements of a software architecture for ICPS and propose a semantic approach to designing ICPS, based on a novel methodology where observation, analysis and adaptation at multiple levels are fundamental to both the *design* and *operation* of such spaces. Our architecture will be designed to:

- Support a powerful model of abstraction that enables access to observations at different levels; this model must be capable of describing QoS needs, exposing errors, failures and uncertainties (of all types) to other levels.
- Support cross-layer specification of functional application logic and non-functional needs (timeliness, information quality) and translation of these multiple QoS needs at the application level to corresponding factors in the infrastructure.
- Enable abstraction layers that have sufficiently well defined languages with well defined semantics such that applications and reasoning at one level can be correctly and consistently translated to the next levels.
- Define and implement mechanisms to enable interdependent system components to share their observed phenomenas, etc.

We envision an overall system architecture of our ICPS systems that is broadly divided into two layers — an application software layer and an infrastructure software layer. The infrastructure software captures all aspects of the connection to the physical space, while the application software manages attributes and properties of applications executing on the physical space. Applications may be written through a higher level event language that allows us to express the application in terms of events of interest as opposed to data that is sensed. The event specification tells us what needs to be done in the application (while the infrastructure software will determine specifics of how this will be done). Connecting the two is the notion of a virtual sensor (which may be a collection of sensor streams and operators on them) that events can tie into to realize the application needs. This two level software system resides on the physical space that contains sensing devices, communication networks, computational entities. The following are features of this two level software architecture for ICPS.

- Each of the two layers has its own hierarchy; the infrastructure layer contains OS, middleware, etc. while the application layer (programmed through an event language) has hierarchical events (with specific semantics).
- Each of the two layers has its multidimensional QoS requirements. At the application layer, this means application timeliness needs and application data quality needs. At the infrastructure layers, we have timeliness constraints, reliability properties and resource limitations which are of course changing over time based on the executing applications. Our system will provide a methodology for translating the application timeliness and data quality needs to QoS parameters at the infrastructure layers (possibly through a quality parameter in a corresponding virtual sensor).
- A novel feature of our approach is that we now support observation and adaptation at each of these layers. While the infrastructure software can monitor (and tune) aspects of the infrastructure, i.e. lower level adaptation; the application layer software will monitor (and tune) aspects of the higher level events. Connecting the two levels of adaptation is a negotiation module that will trigger higher level adaptation when lower level adaptation fails to meet adequate QoS properties.

- A formal modeling and analysis framework supports specification of properties at both levels, including multidimensional QoS properties and the relationships among them. Tuning processes at both the application and infrastructure layers will consult this framework to validate the correctness of the tuning and adaptation factors (sensors, policies, parameters).

3 Multilevel Event-based Framework for ICPS

To enable development of the proposed framework for reconfiguration of Instrumented Cyber Physical Spaces, our approach combines:

- A framework for formal modeling, specification, and reasoning about properties of physical elements (sensor devices, communication components, etc.) and multi-dimensional QoS properties;
- An event-based framework for specifying goals and observations, and relating them; and
- A multi-level observation and adaptation system.

The device models formalized in the modeling framework are used *locally* to tune devices to achieve needed quality of observation under given resource constraints. The interface to the device models allow devices (or collections of devices and services) to be viewed as virtual sensors, by providing information about what observation the device can make, the range of qualities of observation and the resources needed. The event framework provides an application level reasoning/planning system to relate high level events to collections of observables. An observation and adaptation system maintains knowledge about past events and system state, provides a lower level planning system to carry out the primitive observations determined by the event system, and triggers tuning and reconfiguration when the expected multidimensional quality of observation/service parameters are violated. It is a distributed information processing middleware that serves as an abstraction of the sensing, networking and computational resources and also supports interfacing with external components such as databases or human experts. Both levels use the models and reasoning capability of the modeling framework, which in turn provides semantic relationships connecting the levels. Part of the system integration task is to determine the interfaces and possible interactions between these levels and their interfaces to the modeling framework.

It should be pointed out that, abstractly, goals and observations talk about the same thing—events. Therefore, in this particular work, we discuss an event-based framework for specifying goals and observations, and rules relating observations and goals. Goals indicate what is wanted and the quality required, while observations provide evidence for goals. In the event framework, a goal can be considered as a query to determine information about an event. A response provides evidence in terms of observations as well as reasons that the observations are connected to the event of interest as well as a level of confidence. The following tasks capture our approach: (1) the definition of a formal language for specifying events at different levels of abstraction, (2) the development of bottom-up rules that aggregate and abstract observations satisfying lower-level goals to higher-layer observations, and (3) the development of top-down rules to break down high-level goals into subgoals.

There are several challenges to developing a semantic model of events that supports computational inference of real world events from low level observations. These include multiple levels of detail, multiple viewpoints, uncertainty, and synchronization of observations from multiple sources. Although the world is continuous, we work with discrete observations. An observation system may receive low level event streams of a variety of types such as video, audio, temperature, motion (of objects, or seismic), and chemical.

Another challenge is the need to integrate event streams from multiple, distributed sources in a way that accounts both for causal ordering, local ordering, and for the possibility that time stamps from different streams may not be comparable. We assume that primitive events from a single source are linearly ordered according to time stamps. Goal events will have a "causal" relation to the observations making up the evidence. Time stamps can be used to linearly order all events, but this can lead to inconsistencies (for example due to clock skew) and to unnecessary detail. Thus the event framework

will work with event partial orders and intervals, forcing linear order only when necessary. Causal relationships that can be inferred will be used to check consistency of time stamp orderings.

Primitive events include observations made by the infrastructure (including humans) at some place, and at some locally measured time (time interval). They could also be actions carried out by the infrastructure. For example, changing device configuration (e.g., angle, fidelity, etc.), eventually activating switches, door locks, ..., changing location of a mobile sensor. Furthermore, from the event system view point primitive events could correspond to observations recorded in external databases, such as class schedules, class rosters, information about student organizations, medical records, and so on. Sources of primitive events are formalized as *virtual sensors* and form the interface between the event framework and the observation/adaptation system using the shared language provided by the modeling framework.

A goal is an event or a set of events together with a specification of the desired quality of observation. The event framework will use the formal language and rules provided by the modeling framework for expressing data quality at the application and mapping these requirements to properties of virtual sensor observations. Our language will be designed to express multidimensional QoS requirements and compositions, e.g., in terms of priority orderings or weighted metrics.

The proposed event framework will provide general mechanisms for specifying higher level events and rules for them when a set of observations provides evidence for an event. The rules concerning events (together rules for reasoning about observation quality) can be used top down to refine goals into subgoals (as conjunctions or disjunctions). They can also be used bottom up to abstract observations with their associated evidence and confidence to higher level observations. They can also be use to add observations supporting a goal to increase the confidence.

The framework will be prototyped using the Maude rewriting logic environment [1, 4]. In this way the essential features of events and their relationships can be formalized independent of a specific application domain. Rewrite rules will be used to formalize relationships between events.

As a simple example of how a high level event might be representing in the proposed framework, consider determining if there is a person in a given room at a given time (or time interval). We specify an event constructor, `PersonInRoom`, that takes a room and a time as parameters, and two rules specifying observations that could be used to detect such an event (using Maude-like syntax). We assume virtual sensor types `Camera` and `BadgeReader`. Properties of specific instances are associated with the sensor identifiers.

```
op PersonInRoom :  Room Time -> Event .

rl[PersonDetected1]:
   id:Camera S:Snapshot at T:Time and
   I:ImageProcessor says PersonIn(S:Snapshot)
=> PersonInRoom(P:Person,R:Room,T0:Time)
if LocatedIn(id:Camera,R:Room)
with confidence Phi1(id:Camera,
       P:CameraSettings,T0:Time-T:Time, ...)
```

```
rl[PersonDetected2]:
   id:BadgReader B:Badge at T:Time
=> PersonInRoom(P:Person,R:Room,T0:Time)
if AtEntrance(id:BadgeReader,R:Room)
with confidence Phi2(id:BadgeReader,
                     T0:Time-T:Time, ...)
```

A goal might take the form `Goal(PersonInRoom(B175,now), Confidence High)`. The event system would use the rules in the context of declared virtual sensors to attempt to map the goal to observation requests. For the first rule, confidence reasoning (specified in the modeling framework) might refine High confidence to a requirement on the time elapsed between the observation and `now`, and that the Camera settings are in a given range. The resulting observation requirements would then be turned into goals/queries to be satisfied by the infrastructure level observation system.

In addition to specific rules such as the above, there will be rules for combined different observations for the same goal as another way to increase confidence. The example rules use only one level between high-level events and primitive events observed by virtual sensors. In general there may be several levels.

The `PersonInRoom` event could be elaborated to include a person (identifier) as an argument, or additional features of the person, for example height and pose. A goal could use the more complex form but leave some of the parameters unspecified:

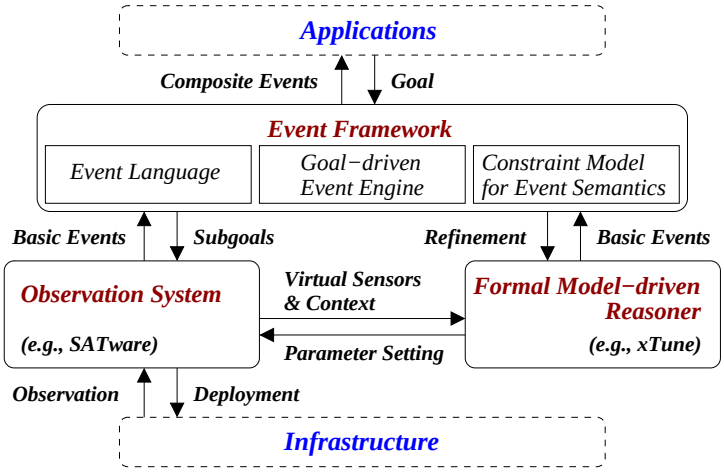

Figure 1: System Prototype

```
Goal(PersonInRoom(P:Person,B175,H:Height,Standing,now) where H:Height < 6 feet, Confidence
High)
```

The resulting queries give flexibility to the infrastructure observation system, depending on the mode of interpretation, i.e., for which variables are values wanted and for which ones the value is not required.

Our overall approach will be hierarchical. For instance, a more general high-level goal is to track the position of a person across the campus and all its buildings, which could involve a reusable identification subgoal such as the one discussed above. Various levels of precision are conceivable, e.g., precision in terms of physical coordinates, or relative to reference frames like buildings or rooms. The tracking frequency can then be determined based on the required level of confidence and precision, as well as background knowledge, e.g., a distribution of or bounds on the velocity for typical movements of people on the campus.

4 Integration into Prototype

To demonstrate the validity of our approach, we will couple the event-based framework with our previous research efforts (SATware [5] and xTune [6]) as depicted in Figure 1. First, users develop applications by describing their application logic and quality needs in the event language supported by an event framework. The event framework translates user goals into high level virtual sensor observations using knowledge about the system and its constraints (as captured in the formal specification framework).

Then, the observation system SATware [5] provides a meaningful view of a cyber-physical region. That is, SATware monitors both the semantic entities in a given region and the infrastructure entities. The semantic entities are comprised of people, objects, and space. These entities and their related basic events are of interest to the application. Examples of basic events are the location of a person, the status of a recycle bin, and the number of people in a room. The infrastructure entities are, on the other hand, the resources that enable the detection of the basic events. Examples of infrastructure entities are routers, communication links, sensing devices, and computing devices. The SATware system monitors events regarding both semantic and infrastructure entities.

Semantic events are forwarded to the event framework and to the formal model-driven reasoner. To the event framework, SATware forwards the basic semantic events that the event framework registers for. SATware can detect a basic event using different virtual sensors. Each virtual sensor is composed of a graph of simple operators (that is, functions) and it can detect a simple basic event at a specific cost and quality [2]. For example, a virtual sensor that includes a face-detection operator and a virtual sensor that includes a finger-print matching operator can detect who enters the building at a different

cost and quality.

In order to decide which virtual sensors should SATware use to detect the basic events, SATware forwards the basic semantic events to the formal model-driven reasoner xTune [6]. xTune, given the current semantic context, decides which virtual sensors SATware should instantiate and with which parameters. For example, on a day-to-day basis one can afford detecting people's location with a cheaper virtual sensor that provides a low degree of accuracy; however, if a building is on fire, one wants to detect the building occupant's location with a high degree of accuracy. Specifically, xTune utilizes the executable nature of Maude specification to conduct formal prototyping and subsequent analysis that later enables a cross-layer system policy/parameter tuning [3].

Once SATware knows which virtual sensors to instantiate, SATware deploys them by selecting in which device each operator executes. For example, image processing operators most likely would be deployed in powerful devices that are in the same subnetwork as the cameras. Other optimizations of the deployment includes if two or more virtual sensors need to instantiate the same operator, the operator might only be instantiated once and shared by both virtual sensors. Once the operators have been deployed, SATware monitors the infrastructure resources and decides to re-deploy the operators (that is, move them to an other device) if the resources change. In addition to the infrastructure entities, SATware monitors the semantic entities and forwards any relevant changes to xTune, which decides if the currently instantiated virtual sensors need to be swapped.

5 Concluding Remarks

This paper presents the challenges and our approaches as the first phase of a project to develop an event-based framework that can customize the operations of instrumented cyber physical spaces (ICPS) based on an observe-analyze-adapt philosophy. We have identified research thrusts and presented an event-based framework for specifying/relating goals and observations, followed by our integration plan for future ICPS. Ongoing and future work in this project includes:

- modeling and specification of physical infrastructure components and multi-dimensional QoS properties;
- A multi-level observation and adaptation system including design of the ICPS reconfiguration framework that deploys monitoring and adaptation decisions to the physical space.

References

[1] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, Jose Meseguer, and Carolyn Talcott. All about maude, a high-performance logical framework. *Lecture Notes in Computer Science*, 4350, 2007.

[2] Bijit Hore, Hojjat Jafarpour, Ramesh Jain, Shengyue Ji, Daniel Massaguer, Sharad Mehrotra, Nalini Venkatasubramanian, and Utz Westermann. Design and implementation of a middleware for sentient spaces. In *Proceedings of ISI'07*, 2007.

[3] Minyoung Kim, Mark-Oliver Stehr, Carolyn Talcott, Nikil Dutt, and Nalini Venkatasubramanian. Constraint refinement for online verifiable cross-layer system adaptation. In *DATE '08: Proceedings of the Design, Automation and Test in Europe Conference and Exposition*, 2008.

[4] Maude System. `http://maude.csl.sri.com`.

[5] SATware Project. `http://satware.ics.uci.edu`.

[6] xTune Framework. `http://xtune.ics.uci.edu`.