Formally Characterizing Device Ecology Workflows with Predictable Observable Effects

Seng W. Loke [1]

School of Computer Science and Software Engineering
Monash University
VIC 3145, Australia

Abstract

We envision a computing platform of the 21st century that takes the form of device ecologies consisting of collections of devices interacting synergistically with one another, with users, and with Internet resources. These devices will perform tasks and work together perhaps autonomously but might need to interact with the user from time to time. We consider *device ecology workflows* as a type of workflow describing how the devices work together. It would be ideal if one can model the devices in a computer and analyze the effects when such workflows are executed in the device ecology. However, we cannot assume that we know much about the details of each device in order to predict and analyze their behaviours. This paper provides a formalization of what it means for a device ecology workflow to have predictable observable effects. This characterization is based on the assumption that the devices in the device ecology are also predictable in a similar sense. The use of this work is as a step towards formal reasoning and analysis about device ecologies, which are normally only constructed ad hoc and using informal software engineering techniques.

Key words: device ecologies, device ecology workflows, observable, labelled transition systems

1 Introduction

The current Internet and networking technologies are enabling smart devices to communicate with one another and with Internet or Web resources. The devices might need to interact effectively in order to accomplish desirable effects for their user(s), and such interaction can occur across the living room or across continents.

[1] Email: swloke@csse.monash.edu.au

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

The American Heritage Dictionary defines the word "ecology" as "the relationship between organisms and their environment." We envision a computing platform of the 21st century that takes the form of device ecologies consisting of collections of devices (in the environment and on users) interacting synergistically with one another, with users, and with Internet resources, undergirded by appropriate software and communication infrastructures that range from Internet-scale to very short range wireless networks. These devices will perform tasks and work together perhaps autonomously but will need to interact with the user from time to time.

There has been significant work in building the networking and integrative infrastructure for such devices, within the home, the office, and other environments and linking them to the global Internet. For example, AutoHan [17], UPnP [12], OSGI [10], Jini [18], and SIDRAH (with short range networking and failure detection) [4] define infrastructure and mechanisms at different levels (from networking to services) for devices to be inter-connected, find each other, and utilize each other's capabilities. Embedded Web Servers [2] are able to expose the functionality of devices as Web services. Embedding micro-servers into physical objects is considered in [14]. Approaches to modelling and programming such devices for the home have been investigated, where devices have been modelled as software components [6], as collections of objects [1], as Web services [11], and as agents [16,3]. However, there has been little work on specifying at a high level of abstraction (and representing this specification explicitly) how such devices would work together at the user-task or application level, and how such work can be managed.

Our earlier work [9] explored the workflow (which we call *device ecology workflows*) as a metaphor for thinking about how collections of these devices (or devices in a device ecology) can work together to accomplish a purpose. It would be ideal if one can model (and simulate aspects of) the devices in a computer and analyze the effects when workflows are executed in the device ecology. Such analysis is useful in order to predict the effects of workflows before they are actually executed (e.g., to see if too much resources will be used or undesirable effects will occur) which will then feedback into the design of the workflow. However, in order to predict the behaviours of these workflows, we might require detailed knowledge of the device. Realistically, we can only assume some knowledge about the device such as some of their observable properties.

This paper presents a formalization of what it means to say that the behaviour of devices and device ecology workflows are predictable. Our approach is based on Labelled Transition Systems (LTS) [13] as a basis for predicting behaviours. We show how to formally relate what is predictable of a device ecology workflow to what is predictable of the devices within the device ecology. Our formalization captures the intuition that the more predictable each individual device is, the more predictable the workflows in the device ecology will be. We also briefly describe analysis that can be done when device ecology

workflows are predictable.

The next section explains the notion of workflows in device ecologies. Then, §3 explains how we define predictability of devices and device ecology workflows. We conclude in §4.

2 Workflows in Device Ecologies

The environment of a device is not only other devices but also human users and the Web resources that it can connect to. We consider several examples of devices working together and interacting with Web resources (e.g., Web services).

Mentioned in the UPnP whitepaper is a script that, on detecting that a master switch has changed to "on", will turn the heater on to a preset temperature, start the answering machine playing new messages, turn the stereo system on set to a favourite station, raise the window blinds, turn the TV to the news station, and turn on the light in the foyer. In this case, the devices effectively work together, as orchestrated by some central coordinator, in order to create the right atmosphere and provide the right services (of informing the user of the new phone messages and news), even though the individual devices might not be aware of this global goal.

Devices might interact directly with one another and with the user. An example is quoted from Berger [2] concerning Thalia appliances, short for Thinking and Linking Intelligent Appliances: "When you set your Thalia alarm clock to wake you at a certain time, it will notify the coffee maker to adjust the time for your morning cup of java. The alarm clock will let you know if you forgot to put the water in the coffee maker. It will also tell the blanket with a brain when to turn off. In the morning, the alarm clock will greet you with the current news and weather. As you are making the pancakes, the kitchen console will automatically adjust the recipe for the number of portions you need. Your HomeHelper kitchen console will store shopping lists, calendars, and telephone numbers that can be downloaded to your HandHelper PDA." The phone might turn the volume of a television down when the user receives an incoming call. Also, an indoor positioning system can be consulted in order to select the most appropriate screen to show news.

Smart appliances might interact with applications and services across the Internet. For example, the fridge can order food that has run out by connecting to a Web service or negotiate with other appliances about resource (e.g., power and networking) consumption. A smart bookshelf can know what books are on its shelf and receive updates on interesting books by connecting to a bookshop Web service or connect to a neighbours smart bookshelf (and even order books based on a user profile and a budget shared with the smart medicine cabinet).

[2] http://www.aarp.org/computers-features/Articles/a2002-07-10-computers_features_appliances.html

Devices might seek human approval for more critical tasks. An automatic lawn sprinkler can check the weather forecast on the Internet to see if it needs to do its watering, and an umbrella after receiving its daily weather forecast can warn of an approaching shower if it recognizes that a particular pair of shoes is leaving it behind. Cars entering particular wireless network cells can begin to connect to electronic services in the area to, for example, get local information of interest to the user and suggest that the user bring an umbrella when leaving the car, have the cars fuel tank be queried by nearby service stations about the fuel level and receive special fuel offers, or find the nearest and cheapest parking spots.

Devices can work together with other devices, the user or Web resources in accomplishing its goals, either as initiated by users or by proactive smart devices. Such workflows might or might not involve devices, users and Web resources, depending on the application semantics. The simpler workflows might involve only one device or two devices, but larger workflows can involve a larger number of devices, as we saw in the examples above.

As an illustration, we consider a workflow device ecology workflow involving a television, a coffee-boiler, bedroom lights, bathroom lights, and a news Web service accessed over the Internet. Figure 1 describes this workflow. The dashed arrows represent sequencing, the boxes are tasks, the solid arrow represents a control link for synchronization across concurrent activities, and free grouping of sequences (i.e., the boxes grouped into the large box) represents concurrent sequences.

This workflow is initiated by a wake-up notice from Janes alarm clock which we assume here is issued to the Device Ecology Workflow Engine (which we call the Decoflow Engine) when the alarm clock rings. This notice initiates the entire workflow. Subsequent to receiving this notice, five activities are concurrently started: retrieve news from the Internet and display is on the television, switch on the television, boil coffee, switch on the bedroom lights, and switch on the bathroom lights. Note the synchronization arrow from Switch On TV to Display News on TV, which ensures that the television must be switched on before the news can be displayed on it. After all the concurrent activities have completed, the final task is to blink the bedroom lights, in order to indicate to Jane that the workflow tasks have completed. This scenario was inspired by and extends that by Berger. This workflow can be described using BPEL4WS and executed using a corresponding workflow engine as outlined in [9].

3 Characterising Device Ecology Workflows with Predictable Observable Effects

Device ecology workflows can be decentralized (ad hoc and peer-to-peer between devices), coordinated by a central engine or hybrid, initiated by a user or a device. Regardless of the architecture or the language used to express the

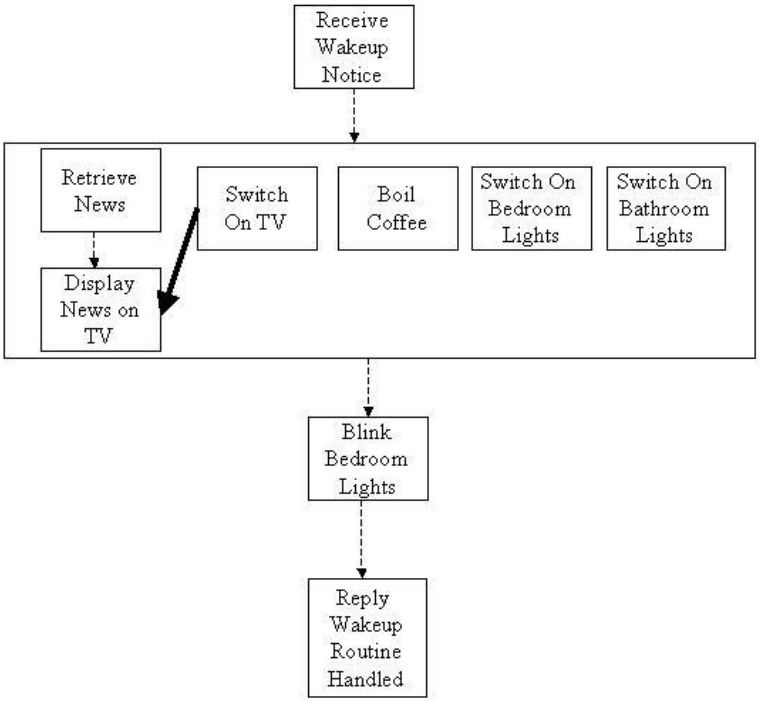

Fig. 1. An Example Device Ecology Workflow.

workflow, we have, in essence, a workflow and a device ecology in which the workflow is executed. Inspired by Hoare's view of programs [5], we consider the execution of workflow in a device ecology as having effects observable via values of a set of attributes. We assume that the values of the variables are observed before the workflow executes and after it terminates (we consider later the situation when the values are observed at intermediate points in time).

3.1 Describing Observable Effects of Device Ecology Workflows

For example, suppose we have a device ecology comprising a table lamp, a room light, and a televsion (or tv). The initial state θ where the table lamp is off, the room light is on and the television is on channel 7 can be described as follows:

$$\theta : \ (< table\_lamp = off >, < room\_light = on >, < tv = on; channel = 7 >)$$

After a workflow terminates, we can represent the final state θ' of the table lamp being on, the room lights remaining on, and the tv status unchanged, as follows:

$$\theta' : \ (< table\_lamp = on >, < room\_light = on >, < tv = on; channel = 7 >)$$

More generally, the observable state of a device is represented by a vector of attribute-value pairs, where the attributes refer to observable properties of the device. The choice of attributes might depend on the application at hand and so, it is meant to be an abstraction of the device. Each device will also have an internal state - we are only concern with the observable properties of the devices here. Moreover, we assume that each device has a set of operations which can be performed on it, each of which might update the device's observable state (and internal state). Given a device d, let $opns(d)$ denote the set of operations which can be performed on it. Note that this can also be seen as an abstraction mechanism - the set of operations we consider might only be those relevant to an application at hand. One way to view this is that each device offers a collection of Web services which can be invoked by a workflow. We also assume that each operation on a device is *introvertive* in that it only affects the device's own internal state and own observable state only. An operation is *extrovertive* if the operation on the device also affects the observable states of other devices.

Given that a device ecology D comprises n devices, the observable state of the device ecology is represented by a tuple $(\bar{d_1} = \bar{v}_1, \ldots, \bar{d_n} = \bar{v}_n)$ of n vectors representing the observable states of the devices, i.e. $\bar{d_i} = \bar{v}_1$ represents the observable state of device i. For short, we shall sometimes use $(\bar{v}_1, \ldots, \bar{v}_n)$, leaving out the attributes. The set of all operations that can be performed on the device ecology, which we denote by $opns(D)$, is given by $\bigcup_{i \in \{1,\ldots,n\}} opns(d_i)$.

3.2 Describing Device Ecology Workflows

In the discussions which follows, we shall assume a device ecology workflow is expressed in a variant of the Web service workflow language DysCo [15], which is valid if we assume that each device can be operated on via invocation of Web services. The EBNF syntax of this language is as follows:

$$
\begin{array}{lll}
W & ::= \epsilon & \text{empty workflow} \\
& \mid T_i^o & \text{task (e.g., Web service invocation)} \\
& \mid W \circ W & \text{sequence} \\
& \mid W + W & \text{choice} \\
& \mid W \mid\mid W & \text{concurrency}
\end{array}
$$

For simplicity, we have considered choice without a condition. In describing a task T_i^o above, given a device ecology D of n devices, o is an operation on a device which can update its observable state (and possibly the internal state), and i identifies a device. For a workflow W, we write $opns(W)$ to denote all the operations that have been mentioned in W.

Each workflow also has an internal state representing a temporary store of results from operations and as a means to provide inputs for operations. Such an internal state might be stored with a centralized workflow engine or

within the device orchestrating the workflow. An operational semantics of all the operators are given in [15] which shows how to evaluate a given workflow expression.

Each step of the execution of the workflow transforms the observable state of the device ecology, i.e. given a device ecology comprising n devices, the workflow W starts execution with an initial device ecology state $\theta_0 = (\bar{v}_1^0, \ldots, \bar{v}_n^0)$, then after termination of the workflow, the final device ecology state is $\theta_k = (\bar{v}_1^k, \ldots, \bar{v}_n^k)$, where k is the number of intermediate steps (each step denoted by $\rightarrow$ and is executed according to the operational semantics of the workflow). Also, let $\theta_0 \stackrel{(w_0,W)}{\rightarrow} \theta_k$ denote that θ_k is an observable state reached after W terminated starting from θ_0 and internal state w_0.

3.3 Predictable Device Ecology Workflows

The question we seek to answer is the following: can a workflow (with a given internal start state) be executed in the device ecology, terminating with desired effects, where desired effects might be expressed as a property on the final observable state?

One way to solve this problem is to consider a simulation of the execution. But in order to perform a simulation, one must have a model of each device. In general, it is difficult to predict the effects of a workflow if we do not know much about the devices. But we assume that each device in a device ecology is a grey box, i.e., the device has an observable state and an internal state. Since the internals of a device is hidden (which is realistic to assume), it might not be possible to fully determine the behaviour of the device after a workflow's operation. However, if we assume that each device is predictable in the sense that we describe later, then possibly, the effects of the workflow on the device ecology might also be predictable in some sense. Below, we make this two senses precise, which is in terms of representability of their behaviours via an LTS.

We first introduce a criteria of predictability for each device, where we assume each device has a set of operations on it.

Definition 3.1 A device with a finite set of operations O is *externally predictable* iff its observable behaviour can be characterized by a labelled transition system L over O given by a pair (S, T) where S is the set of observable states of the device (where a vector of attribute-value pairs represents an observable state of the device), and T is a ternary relation where $T \subseteq (S \times O \times S)$, and T covers all cases in the sense that $\forall s \in S$ and $o \in O$, there is some $s' \in S$ such that $(s, o, s') \in T$.

In the above, we formalize "predictability" as the existence of an LTS which would allow the device's observable behaviour to be predicted, and "external" is in the sense that the LTS is based on observable states of the device (without knowing details of the internals of the device). Thus, if L describes

the observable behaviour of a device, then given that one observes that the device is in some (observable) state, one can predict what observable state the device will go to next after an operation, by using L. Note that L is nondeterministic in that there might be more than one possible next states. Moreover, we have assumed that operations do not have parameters. Realistically, some would have. For example, operations $o(input)$ and $o(input')$ use the same operation o but with different inputs, and so might result in different transitions. In such cases, where it is possible to enumerate the operation with different inputs, i.e. we treat $o(input)$ and $o(input')$ as different operations in O, and the above definition can be used without change. In cases where such an enumeration is not finite, we can use nondeterminism to model the operations. For example, if whatever the inputs to o starting from s, the device only goes to s, s' or s'', then the behaviour of o can be represented by (s, o, s'), (s, o, s'), and (s, o, s''). With this, the above definition is applicable for operations with whatever inputs.

We introduce a corresponding definition for the predictability of a workflow executed in a given device ecology.

Definition 3.2 Given a device ecology (i.e., a set of n devices) $\mathcal{D}$ and a device ecology workflow W, where $opns(W) \subseteq opns(D)$ (i.e. W doesn't require an operation that does not exist in D) with initial internal state w_0, written (w_0, W), (w_0, W) is externally predictable with respect to D iff there exists a labelled transition system $L_{(w_0,W)}$ over $\{(w_0, W)\}$ given by a pair $(\mathcal{S}, \mathcal{T})$, where $\mathcal{S}$ is the set of observable states of the device ecology (each observable state of the device ecology is given by a tuple of n vectors), and $\mathcal{T}$ is a ternary relation where $\mathcal{T} \subseteq (\mathcal{S} \times \{(w_0, W)\} \times \mathcal{S})$, and $\mathcal{T}$ covers all cases in the sense that $\forall\theta \in \mathcal{S}$, there is some $\theta' \in \mathcal{S}$ such that $(\theta, (w_0, W), \theta') \in \mathcal{T}$.

Note that the only action in $L_{(w_0,W)}$ is the workflow (w_0, W). So, a workflow (w_0, W) is externally predictable with respect to D means that regardless of what observable state the device ecology is currently in (say $\theta \in \mathcal{S}$), if (w_0, W) is started in state θ, then one can consult $L_{(w_0,W)}$ to see what observable state the device ecology will be in after (w_0, W) terminates, having started in θ.

Now we prove the following theorem.

Theorem 3.3 *Given a device ecology D, a workflow (w_0, W), where $opns(W) \subseteq opns(D)$, is externally predictable with respect to D iff each device in D is externally predictable.*

Proof. Given a device ecology D comprising n devices, we prove the theorem by showing how to construct $L_{(w_0,W)}$ for any (w_0, W) as follows. Let $L_{(w_0,W)} = (\mathcal{S}, \mathcal{T})$, for some $\mathcal{S}$ and $\mathcal{T}$ which we will construct below. Since each device in D is externally predictable, for device $i \in \{1, \ldots, n\}$, we have its LTS denoted by (S_i, T_i). $\mathcal{S}$ can then be given by the Cartesian product $S_1 \times \ldots \times S_n$. Let $\theta = (\bar{d}_1 = \bar{v}_1, \ldots, \bar{d}_n = \bar{v}_n) \in \mathcal{S}$, i.e. each vector of pairs $\bar{d}_i = \bar{v}_i \in S_i$. We

need to derive $\theta' \in \mathcal{S}$ such that $\theta \overset{(w_0,W)}{\rightarrow} \theta'$. We show that, in fact, there are many possibilities for θ'.

For a device i, in W, there might be operations on device i and there might be operations on other devices. After (w_0, W) has terminated, device i would have moved through a number of observable states according to what operations in W were on device i, and since we have (S_i, T_i), we can compute the observable states of device i after (w_0, W) has terminated. In other words, we want to show that we can derive one or several possible $\bar{v}_i'$ such that $\bar{v}_i \overset{(w_0,W)}{\rightarrow} \bar{v}_i'$ using (S_i, T_i). We show this by induction on the syntax of workflows as follows.

(i) If $W = T_i^o$, then we have $\bar{v}_i \overset{(w_0,T_i^o)}{\rightarrow} \bar{v}_i'$, where $(\bar{v}_i, o, \bar{v}_i') \in T_i$. Since we might also have $(\bar{v}_i, o, \bar{v}_i'') \in T_i$ where $\bar{v}_i' \neq \bar{v}_i''$, we actually have a set of possibilities. Denoting this set by F_i, we write $\bar{v}_i \overset{(w_0,W)}{\rightarrow} F_i$ to mean the workflow W brings the observable state to the possible states in F_i.

(ii) If $W = W' \circ T_i^o$, and suppose that $\bar{v}_i \overset{(w_0,W')}{\rightarrow} F_i$, then for any $s \in F_i$, and suppose w_1 is the internal state after W', by (1) we have $s \overset{(w_1,T_i^o)}{\rightarrow} F_i^s$. So, we have $\bar{v}_i \overset{(w_0,W)}{\rightarrow} \bigcup_{s \in F_i} F_i^s$.

(iii) If $W = W' + W''$, and suppose that $\bar{v}_i \overset{(w_0,W')}{\rightarrow} F_i'$ and that $\bar{v}_i \overset{(w_0,W'')}{\rightarrow} F_i''$. Then, we have $\bar{v}_i \overset{(w_0,W)}{\rightarrow} F_i' \cup F_i''$.

(iv) In the case of $W = W'||W''$, without lost of generality, we can consider the special case $W = T_i^{o1}||T_i^{o2}$. In such a case, we assume that the device handles the concurrency by serializing the operations. Hence, $T_i^{o1}||T_i^{o2}$ is interpreted as either doing $o1$ first and then $o2$ or in the reverse order, i.e.

$$T_i^{o1} \;||\; T_i^{o2} \quad = \quad (T_i^{o1} \circ T_i^{o2}) \;+\; (T_i^{o2} \circ T_i^{o1})$$

And so, from (1), (2) and (3), we can work out some F_i such that $\bar{v}_i \overset{(w_0,W)}{\rightarrow} F_i$.

Now, we can do the above for all the devices, and since the operations are introvertive, we can construct θ' by forming the tuples of final observable states (by Cartesian product) from the constituent devices' final observable states, namely, $\theta' \in F_1 \times \ldots \times F_n$, where for each i, $\bar{v}_i \overset{(w_0,W)}{\rightarrow} F_i$. In other words, for the given θ, we have a set of tuples

$$\{(\theta, (w_0, W), \theta') \mid \theta' \in F_1 \times \ldots \times F_n\}$$

where the F_is are as defined above. □

As an example, we consider a device ecology with devices $d1$, $d2$ and $d3$. Suppose all the devices are externally predictable. Then, let the LTSs capturing the observable behaviours of the devices be given as follows:

```
d1: S1 = {s1(1), s1(2)}
    O1 = {o1(1)}
    T1 = {(s1(1),o1(1),s1(2))*, (s1(2),o1(1),s1(1)),
          (s1(2),o1(1),s1(2))}
d2: S2 = {s2(1), s2(2), s2(3)}
    O2 = {o2(1)}
    T2 = {(s2(1),o2(1),s2(2)), (s2(2),o2(1),s2(1))*,
          (s2(3),o2(1),s2(2))}
d2: S3 = {s3(1), s3(2)}
    O2 = {o3(1), o3(2)}
    T3 = {(s3(1),o3(1),s3(2)), (s3(2),o3(1),s3(1)),
          (s3(1),o3(2),s3(2))*, (s3(2),o3(2),s3(2)),
          (s3(1),o3(2),s3(1))}
```

Then, for a given start observable state of the device ecology such as `(s1(1),s2(2),s3(1))`, and workflow `o1(1).(o2(1)+o3(2)).o3(2)`, we can work out the (final) observable state after the workflow completes by following the above transitions. In this case, a possible final state is `(s1(2),s2(1),s3(2))` (when `o2(1)` was chosen) using the transitions marked `*`.

3.4 Analyzing Device Ecology Workflows

Given a device ecology D, and that a workflow (w_0, W), where $opns(W) \subseteq opns(D)$, is externally predictable with respect to D, we can then start to analyse the properties of W with respect to D. For example, we can define preconditions, postconditions, and invariants. Such assertions will be predicates over the values of the attribtes $(\bar{d}_1, \ldots, \bar{d}_n)$. We can also define an *invariant* I where $\theta \vDash_C I$ for any θ, where $\theta \vDash_C P$ means the assignment of values to attributes in θ satisfies the predicate P via some procedure C. The invariant might be a measure on the device ecology such as number of lights turned on at any given point of the workflow execution - we might have the invariant $num\_lights\_on < 20$, where C is a procedure that processes the values in θ to compute the value of num_lights_on. A similar relation can be defined with procedures C' and C'', and predicates for precondition P and postcondition Q: $\theta \vDash_{C'} P$ and $\theta' \vDash_{C''} Q$, where $\theta \overset{(w_0,W)}{\rightarrow} \theta'$. Given such assertions, workflows can be compared similar to programs, but we can use this mechanism to characterise properties of device ecologies.

Robustness analysis is useful when one wishes to examine the capabilities of a new device ecology obtained by adding new devices to the existing device ecology, removing particular devices, or replacing particular devices. We envision a future where when one (say Jane) who has a device ecology D at home buys a new device d, he or she would like to know what new workflows can execute on the new device ecology $D \cup \{d\}$, whether the new device d can work with D, or what observable effects of a workflow executed in an updated device ecology $(D \cup \{d\}) \backslash \{e\}$ where d replaces e. Moreover, Jane might also want to know if a workflow can still be executed after a device

has been removed from the device ecology. Such analysis can be done using some abstraction of the devices, and in this case, it is the LTS describing the observable behaviour of devices.

4 Conclusion and Future Work

We have provided a formalization in terms of the existence of an LTS of what it means for a device ecology workflow to have predictable observable effects. This characterization is based on the assumption that the devices in the device ecology are also predictable in a similar sense. Theorem 1 formalizes the relationship between the predictability of a device ecology workflow and the predictability of the devices. An implication here is that if the LTSs of the devices are very detailed, then an LTS for a device ecology workflow can be constructed which is also very detailed. Hence, our formalization captures the intuition that the more predictable the devices in a device ecology are (or the richer or larger their LTSs), the more predictable a workflow executed in the device ecology will be (i.e., the richer or larger its LTS).

This work is an initial step towards a formal theory of predictable device ecology workflows, with myriad practical uses. Based on such mechanical analysis, smart devices or users can check the effects of workflows they intend to initiate. A user might keep a store of workflows which can be checked just before invocation. The real world is, of course, not quite fully predictable, even with nondeterminism to represent possibilities (since there will be unforeseen possibilities) but we seek to provide a level of predictability for the collective behaviour of devices. Interactive simulation can be used to fine-tune workflows so that they produce desired effects and to study such workflows for different scenarios of devices. We are currently building a grahpical simulator for device ecology workflows. A more comprehensive study of properties of workflows can be done using model checking (e.g., [7]).

Further work will consider workflows with operations not only on devices but also general Web services and operations that ask the user questions, workflows with conditional choice, workflows with synchronization for concurrency, and devices with extrovertive operations which can be modelled as dynamically extensible workflows. This means that when an extrovertive operation is invoked on the device, the workflow is augmented with further operations that capture the effects of the operation on other devices. Dynamically extensible workflows might become non-terminating - we will need to explore conditions that lead to terminating dynamically extensible workflows. We will also explore automated means of constructing $L_(w_0, W) = (\mathcal{S}, \mathcal{T})$ given the LTS of the devices observable behaviours. We note that the problem can become intractable with a large number of states. Experimentation is needed to understand the sizes of problems tractable. However, we recall that the LTS model of devices is an abstraction and need only capture relevant aspects of the device that are to be analyzed. Moreover, the relation $\mathcal{T}$ can perhaps

be intensionally defined via an algorithm instead of extensionally detailed. Given all that, for ad hoc and unanticipated behaviours, we see the limitations of our theory for ad hoc workflows and on-the-fly decisions that devices might make (though nondeterminism in a workflow supports representation of (albeit) foreseen possibilities). For scalability, we can consider a device-and-conquer and abstraction approach where a workflow and device ecology can be partitioned and analyzed separately. Techniques for combining separately constructed LTSs from parts of workflows and device ecologies, and for combining results of analysis can be investigated. Our guess is that sizable workflows and device ecologies can be analyzed with our approach, but admit that different methods are required for larger scale finer granularity collections of complex systems of devices - this will be an avenue of future work once the scalability of our approach has been more precisely defined. Finally, our condition of $opns(W) \subseteq opns(D)$ is a first test to see if a workflow can execute in a device ecology. We admit that this condition can be a lot more elaborate and that semantic matching might be used to match workflow tasks to the actual devices that will carry the tasks out. Other work such as [8] has addressed this issue of selecting devices to be aggregated in order to meet a user request.

References

[1] AHAM, "Connected Home Appliances Object Modelling," AHAM CHA-1-2002, 2002.

[2] Bentham, J., "TCP/IP Lean: Web Servers for Embedded Systems (2nd Edition),", CMP Books, 2002.

[3] Carabelea, C. and Boissier, O., "Multi-agent Platforms for Smart Devices: Dream or Reality?," Proceedings of the Smart Objects Conference (SOC'03), Grenoble, May 2003.

[4] Durand, Y., Vincent, S.P.J.-M., Marchand, C., Ottogalli, F.-G., Olive, V., Martin, S., Dumant, B., and Chambon, S., "SIDRAH: A Software Infrastructure for a Resilient Community of Wireless Devices," Proceedings of the Smart Objects Conference (SOC'03), Grenoble, May 2003.

[5] Hoare, C.A.R., "Unified Theories of Programming," 1994. Available at ftp://ftp.comlab.ox.ac.uk/pub/Documents/techpapers/Tony.Hoare/theory94.ps.

[6] Jahnke, J.H., d'Entremont, M., and Stier, J., *Facilitating the Programming of the Smart Home*, IEEE Wireless Communications, December, (2002), 70-76.

[7] Karamanolis, C., Giannakopoulou, D., Magee, J., and Wheater, S., "Model Checking of Workflow Schemas," Proceedings of the 4th International Enterprise Distributed Object Computing Conference (EDOC), Makuhari, Japan, September 2000.

[8] Kumar, R., Poladian, V., Greenberg, I., Messer, A., and Milojicic, D., "Selecting Devices for Aggregation," Proceedings of the 5th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2003), 2003.

[9] Loke, S.W., "Service-Oriented Device Ecology Workflows," Proceedings of the International Conference on Service-Oriented Computing, Italy, pages 559-574, December 2003, Springer-Verlag.

[10] Marples, D. and Kriens, P., *The Open Services Gateway Initiative: An Introductory Overview*, IEEE Communications, December, (2001), 2-6.

[11] Matsuura, K., Haraa, T., Watanabe, A., and Nakajima, T., "A New Architecture for Home Computing," Proceedings of the IEEE Workshop on Software Technologies for Future Embedded Systems (WSTFES03), pages 71–74, May 2003.

[12] Microsoft Corporation, "Understanding $UPnP^{TM}$: A White Paper" Available at http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc.

[13] R. Milner., *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[14] Nakajima, T., "Pervasive Servers: A Framework for Creating a Society of Appliances," Proceedings of the 1AD: First International Conference on Appliance Design, pages 57–63, May 2003.

[15] Piccinelli, G., Finkelstein, A., and S.L. Williams, S.L., "Service-Oriented Workflows: the DySCo Framework," Proceedings of the Euromicro Conference, Antalya, Turkey, 2003. Available at http://www.cs.ucl.ac.uk/staff/A.Finkelstein/papers/euromicro2003.pdf.

[16] Ramparany, F., Boissier, O., and Brouchoud, H., "Cooperating Autonomous Smart Devices," Proceedings of the Smart Objects Conference (SOC'03), Grenoble, May 2003.

[17] Saif, U., Gordon, D., and Greaves, D.J., *Internet Access to a Home Area Network*, IEEE Internet Computing, Jan-Feb, (2001), 54-63.

[18] Waldo, J., *The Jini Architecture for Network-Centric Computing*, Communications of the ACM, July, (1999), 76-82.