Context-aware Mobile Security

Experimental Validation of Reliable and Secure Context Detection

Christian Jung, Denis Feth
Fraunhofer IESE
Kaiserslautern, Germany
Email: christian.jung@iese.fraunhofer.de, denis.feth@iese.fraunhofer.de

***Abstract*—The exploitation of context-awareness, especially in mobile devices bears a huge potential. For example, mobile workers benefit from systems that adapt security settings or user interfaces to the current situation. However, the correct detection of contexts strongly relies on raw data from various context information sources that might be neither trustworthy nor authoritative. In this work, we present an extension to a context model that explicitly copes with trustworthiness of context information, i.e., its vulnerability to forgery, as well as their conduciveness denoting the source's decisive impact on context detection. Context descriptions based on this model can, for example, be used in security-critical environments to enforce security policies. We show the applicability of our approach in an industrial setting. In addition, we present the results of our experiments with respect to precision and recall of the context detection.**

Keywords*–*Context-Awareness; Context-Modeling; Security

I. Introduction

Context-awareness of software applications is still in its infancy [1], [2], although it has been researched since the beginning of the nineties [3]. Recently, the rise of mobile technologies introduced a new class of devices with various sensors providing context information. For such devices, context-awareness can be particularly useful to adapt user interfaces or security measures to the current situation, in order to relieve the user. For example, context-awareness enables more flexible control by limiting the applicability of security measures only to situations where they are indispensable (e.g., display timeout for mobile devices is set to fifteen minutes in the office, thirty minutes at home, and to two minutes elsewhere).

An important building block for enabling context-aware security is context modeling. The user's contexts (i.e., her current activity and device situation) have to be determined by aggregating low-level contextual information, such as current location, battery consumption, or connectivity of her mobile device. However, in order to provide reliable decision support, context descriptions have to be trustworthy, reliable, and accurate, especially to support security decisions. Thus, the context evaluation must consider information about how easy it is to counterfeit contextual information.

To this end, a methodology for eliciting and modeling contextual information is needed that yields reusable and comparable context descriptions. In particular, this method must support the identification of suitable context information sources and the aggregation of low-level pieces of context information into an overall context description.

Contribution. In [1], we present our context model and context descriptions that explicitly include a relevance and a security rating for each context information source. The two quality attributes are used to improve the recognition accuracy, which is an open challenge in activity recognition [4], [5]. These ratings enable us to provide quality statements for the accuracy of context detections. Security decisions benefit from the quality statements within a context description, aggregated during run-time. We extend [1] by showing the applicability of our approach in an industrial setting. In addition, we present the results of our experiments with respect to precision and recall of the context detection.

Paper Structure. The paper is structured as follows: Our context modeling approach is presented in Section II, followed by Section III addressing uncertainty of context information sources. In Section IV, we apply our approach to an industrial scenario from a large German company. We present our evaluation with respect to the quality of our context detection in Section V. Section VI addresses related work in the area of context definition and context information modeling. Finally, Section VII provides a summary and an outlook on future work.

II. Context Model

Our work uses a combination of existing definitions and descriptions of context, which are further described in our related work section (Section VI). We partition context sources into virtual and physical information sources, depending on the origin of the information, and we logically link them (similar to Hofer et al. [6]). As this work focuses on mobile devices and their users, activities of the user are an important aspect, as well as the operational state of the device. Thus, we define context as:

> *Context is the state of all virtual and physical information sources that characterize the activity of the user and the operational state of the mobile device in a specific situation at a certain time.*

Figure 1 presents a macroscopic view of the core parts of our model and their interrelations. The model mainly describes relations between the user, the mobile device, and the context. The context is broken up in a user context (user-centric context) and a device context (mobile device-centric context). We assume that a user can have one or more mobile devices and that a user has always her own mobile device, which she will not share with other users.

A user can perform several activities in a specific situation at a certain time. The user context depends on the activity and

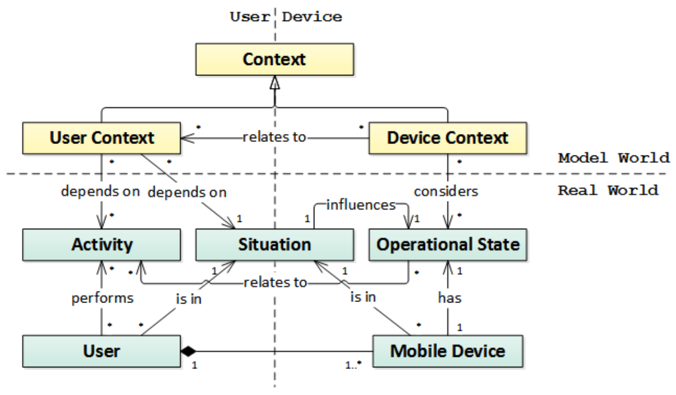


Figure 1. Context Model (Macroscopic Viewpoint).

the current situation of the user. In an ideal world, the user context would include all information about the activities and the situation. Hence, the information in the model world would be congruent with the information in the real world. However, due to technical constraints of the context information retrieval, the detection capabilities (e.g., environment sensors) are limited. In addition, the boundaries between certain activities are fluid and often cannot be determined by context information sources. Accordingly, the differentiation between activity, situation, and user context will persist. Hence, the relations between activity, situation, and user context depends on the granularity of the modeled user contexts. Our context model has to cope with such uncertainties.

A mobile device has an operational state and acts within a specific situation. Hence, the situation of the device includes internal states of the device and attributes of the environment. The overall device context considers the operational state, which may be influenced by the current situation. Similar to the relation between activity, situation, and user context, the device context would always match the current situation and operational state in an ideal world. However, due to technical limitations, several distinct device situations lead to the same device context. Moreover, the operational state may relate to the activity of the user. This relation must not necessarily hold, as the user can also perform device-independent activities. However, for automatic context detection, we can only use the operational state. Finally, the device context also depends on the user context. Again, in an ideal world, the device context would include all information about the situation and the operational state.

It is apparent that the environment and internal states of the device can only be sensed by context information sources that are technically available. Thus, device context and user context must be detectable by existing context information sources, but are only approximations of the real world, neglecting unmeasurable information. In contrast, activity, situation and operational state strive to represent the world as it really is.

The core part of the context model is the context itself. The goal is to model the user and device context as an abstraction and aggregation of pieces of information obtained from various context information sources.

A. *Context Description Structure*

A context description can be a logical or arithmetic expression. The context description aggregates raw sensor data as evaluators and combines them to a tree structure. For example, we can specify the context “LowBattery” as shown in Figure 2.

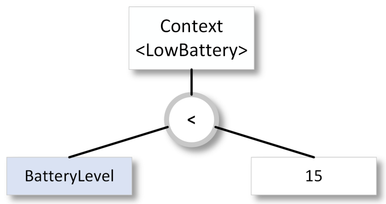


Figure 2. Example Context Tree “LowBattery”

To model contexts, *Expressions* can be combined to form arbitrarily complex descriptions (see Figure 3). Expressions can be combined and nested in a way that the respective overall result is a Boolean value with additional ratings for security and relevance (see Section III). *GenericExpression* forms the basis for all other types of Expressions (arithmetic, comparison, and logic) and a *ConstantExpression* holding a constant value. The Expression interface has a method for evaluating itself and a method for retrieving the return type. For type safety, it is important to have these type assignments, as a context description, at least in theory, could combine any expression type. However, there is a check whether the relation is allowed. For instance, a comparison between a Boolean type and a list of values would be rejected.

A specialization of the GenericExpression is the BinaryExpression, which allows exactly two subordinated expressions. *ComparisonExpressions*, for instance, take exactly two subexpressions for their evaluation.

ArithmeticExpressions are expressions for addition, subtraction, multiplication and division. For evaluating the expression, all assigned sub-expressions are joined by the appropriate operation. In general, ArithmeticExpressions can contain an unlimited number of nested expressions.

Similar to ArithmeticExpression, a *LogicExpression* can take an unlimited number of nested expressions for evaluation. For the moment, *and*, *or*, and *not* with their usual semantics

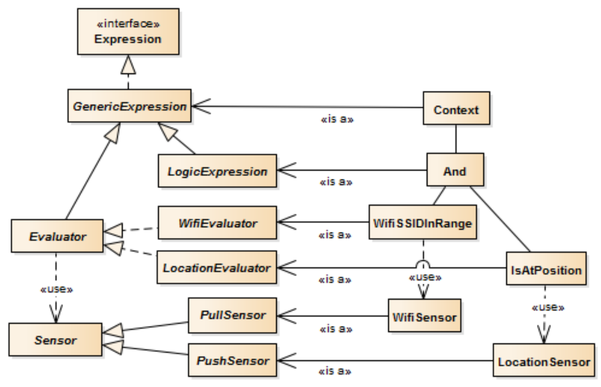


Figure 3. Excerpt of Expression Model.

```
<context id="example-context">
  <logic:and>
    <logic:or>
      <evaluator name="GenericLocation" relevance="5">
        <param name="latitude" value="49.431479"/>
        <param name="longitude" value="7.7520288"/>
        <param name="distance" value="15.0"/>
        <param name="provider" value="0"/>
        <param name="maxAge" value="3600"/>
        <param name="resultType" value="boolean"/>
       </evaluator>
       <evaluator name="WiFiIsSSIDInRange" relevance="3">
         <param name="maxAge" value="60"/>
         <param name="keepEnabled" value="true"/>
         <param name="scheduleInterval" value="15"/>
         <param name="ssid" value="wlan-home"/>
         <param name="resultType" value="boolean"/>
        </evaluator>
    </logic:or>
    <!-- Arithmetic demo: 2*2*4 >= 10+(36/6) -->
    <comparison:greaterEqual>
      <arithmetic:multiply>
        <constant type="double" value="2"/>
        <constant type="double" value="2"/>
        <constant type="double" value="4"/>
      </arithmetic:multiply>
      <arithmetic:add>
        <constant type="double" value="10"/>
        <arithmetic:divide>
          <constant type="double" value="36"/>
          <constant type="double" value="6"/>
        </arithmetic:divide>
      </arithmetic:add>
    </comparison:greaterEqual>
   </logic:and>
</context>
```

Figure 4. Evaluator Example.

have been implemented. Future extensions could include fuzzy logic or other evaluation capabilities.

The *Evaluator* can express various behaviors, for example, aggregation or temporal changes of information. Every functionality that cannot be expressed by arithmetic or logic expressions has to be provided by evaluators. Each evaluator implementation encapsulates a specific operation of *Sensor* data, which means, it is one possible abstraction of the operative behavior of the information source and performs a first aggregation of raw values from the measurement (if needed). Evaluators can be included as sub-expressions at arbitrary locations within the presented structure.

Sensors provide an abstract representation for context information sources, which usually deliver low-level information, such as the current coordinates of the Global Positioning System (GPS) or acceleration values. In general, we distinguish between push and pull behavior, based on the characteristics of the underlying information source. Some sensors can be configured to provide (i.e., push behavior) new information as soon as new data is available (e.g., acceleration or location sensors). Such push sensors usually contain parameters to configure conditions when data updates will be delivered. For example, the location sensor will only deliver new information if the mobile device location changed by at least 50 meters. Other sensor information has to be queried regularly (i.e., pull behavior) to obtain current information (e.g., mobile device settings, calendar). Pull sensors contain a *scheduleInterval* parameter specifying the frequency of data updates. For example, wireless network information can be requested every five minutes.

Our preferred format for a context description is XML (see Figure 4). In general, a configuration can contain several expressions of types arithmetic, comparison, or logic (future extensions could extend the list of available expression types).

The specification in Figure 4 includes two different evaluators: a GenericLocation evaluator for checking a specific location and a WiFiIsSSIDInRange-Evaluator to scan for specific wireless Service Set Identifiers (SSID). The location evaluator consumes the following parameters: location values (i.e., *latitude* and *longitude*), *distance* (specifying the data delivery and distance accuracy), *provider* (location information from network and/or GPS) and *maxAge* (allowed data age for evaluation). The wireless evaluator uses the parameter *ssid* to scan for this specific wireless SSID. In addition, the parameter *keepEnabled* prevents the user from disabling the wireless network sensing. The listing also shows an example of arithmetic expressions and comparisons. Evaluators may contain a relevance and a security rating, which are described next.

III. Dealing with Uncertainty

The basic question when building a context-aware system is: "To which extent is the context detection reliable?" We distinguish three major uncertainties to deal with: trustworthiness, relevance, and accuracy.

Accuracy information is typically already provided by sensors used in mobile systems. Therefore, we focus on other two uncertainties and introduce two quality metrics: a *security rating*, denoting the difficulty for an adversary to counterfeit the measurement of the context information source and a *relevance rating*, expressing the value of the context information source for the identification of the overall context.

For both quality metrics, we have to find suitable thresholds and combine them to trust the context detection in the given usage scenario. For example, a context based on low ranked context information sources only, can easily be counterfeited and might not be trustworthy. Similar, some context information sources are more supportive to detect the current context than others. For example, the wireless network information is well suited to detect the context "home", while it is rather bad for detecting activities such as "running".

A. Security Rating

Every evaluator has a security rating assigned to it. The rating takes the values from one (very low) to five (very high). Basically, the security rating denotes the trustworthiness of the context information, i.e., its vulnerability to forgery. The security rating within our work is defined as follows:

The Security Rating is a global indicator expressing difficulty and challenge for an adversary to counterfeit a context information source.

The following aspects have to be considered, when rating an evaluator. A security expert or group of experts have to perform this task based on an attacker model when an evaluator is developed. The experts have to assess the necessary preconditions for successfully counterfeiting an information source:

(i) insider knowledge or configuration details
(ii) special expertise or knowledge to perform the operation
(iii) special software or application

(iv) special hardware or equipment
(v) influence on information source (backend or environment change)

Based on these prerequisites, the metric to determine the rating for every evaluator is defined as follows:

- **1 (very low)**: 0 out of 5 prerequisites needed
  It is easy to counterfeit the measured values (e.g., just change or enter the value). An example for such a rating is the time of the mobile device or calendar entries of the user.
- **2 (low)**: 1 out of 5, but not prerequisite (iv)
  The manipulation of the sensed value can be done with little effort. An example is to simulate a high light intensity with a torch or to shake the device to forge acceleration values.
- **3 (medium)**: 2 out of 5 OR 1 out of {(iii), (iv), (v)}
  Some preparations are required, but they are not really challenging. An example would be faking the SSID or BSSID of a WiFi hotspot, which can directly be done by using a second smart mobile device.
- **4 (high)**: 3 out of 5 OR 2 out of {(iii), (iv), (v)}
  The required measures are challenging for an adversary, and without special knowledge, it would not be possible to perform the attack. An example is to simulate that the device is connected to an encrypted (mobile) network.
- **5 (very high)**: 4 out of 5 OR 3 out of {(iii), (iv), (v)}
  Forging of context information sources requires deep knowledge about the internal configuration and significant expertise; moreover special equipment is needed, such as software and hardware. An example is the GPS sensor or cell phone tower information, for which an attacker would need special hardware and knowledge how to use it.

The security rating has to be defined once, and it has a global scope for every instantiated context tree. However, it is possible to manually change the rating by explicitly setting it in the evaluator tag of the context description. For example, the security rating of a virtual context information source can vary according to its trustworthiness. A read-only enterprise calendar will be much harder to counterfeit than the personal calendar maintained by the user itself. Hence, we can manually assign a higher security rating to the evaluator using the read-only enterprise calendar.

We rated all our evaluators by asking two security experts from our institute to assess the five preconditions. Figure 5 presents possible answers for their assessment.

✓	The precondition is necessary for a successful manipulation.
✗	The precondition is not necessary for a successful manipulation.
/	The precondition is not important or not relevant for the rating.
-	The precondition is not applicable.
?	The decision cannot be made and needs further discussion.

Figure 5. Possible Answers for Assessing Security Rating Preconditions

In addition, the experts had to write a brief statement for every precondition for being able to reconstruct their assessment and for being used in later discussions with the other security expert. Figure 6 shows the filled table for the *Bluetooth-Evaluator* by one security expert. The security expert rates the *BluetoothEvaluator* with a security rating of 4 (high), as 3 out of 5 preconditions are necessary for a successful manipulation.

BluetoothEvaluator		
✓	(i)	Generally speaking the Bluetooth interface is safe. There are possible attacks in certain configurations but those are only applicable if the user has its Bluetooth device set to be visible. With proper equipment and knowledge it is possible to disguise a device as something else. If the profile of the connecting partner is known it is also possible to disguise as a specific device.
✗	(ii)	The general attack path is to trick the device into thinking it is communicating with a known device and therefor the system takes physical this presence into account for a special context. However this is false since the attacked device is communicating with is just disguised as a known device.
✓|✗	(iii)	Either special hardware or special software is needed to trick the attacked device into thinking it communicates with a known device.
✓|✗	(iv)	Either special hardware or special software is needed to trick the attacked device into thinking it communicates with a known device.
/	(v)	Not applicable.
Security Rating: 4 (high)		

Figure 6. Security Rating Assessment for BluetoothEvaluator

Finally, we collected the information from the two security experts and determined a security rating for every evaluator based on their assessment. For instance, evaluators using the accelerometer values have been rated with very low, as a device user can easily manipulate these values. Contrary, evaluators using Android's location services are rated as very high, as the location services are using a multitude of location sources (i.e., GPS, Cell-ID, Wi-Fi) [7], which are hard to manipulate.

B. Relevance Rating

For each context, we assign a second rating to every evaluator, expressing the contribution of the information source to the overall context identification. Similar to the security rating, it accepts values from one (very low) to five (very high). The rating represents whether the provided information tends to be decisive or has a less authoritative impact on context detection. The relevance rating is defined as follows:

> *The Relevance Rating is a local indicator expressing the correlation of a context information source with an activity, or situation.*

The relevance rating depends on the modeled context, but also on the quality of a sensor and cannot be specified by just following generic guidelines. Thus, the retrieval of the relevance rating is part of an automatic derivation process, which has been described in [8]. This process yields context descriptions that can be used in operative environments.

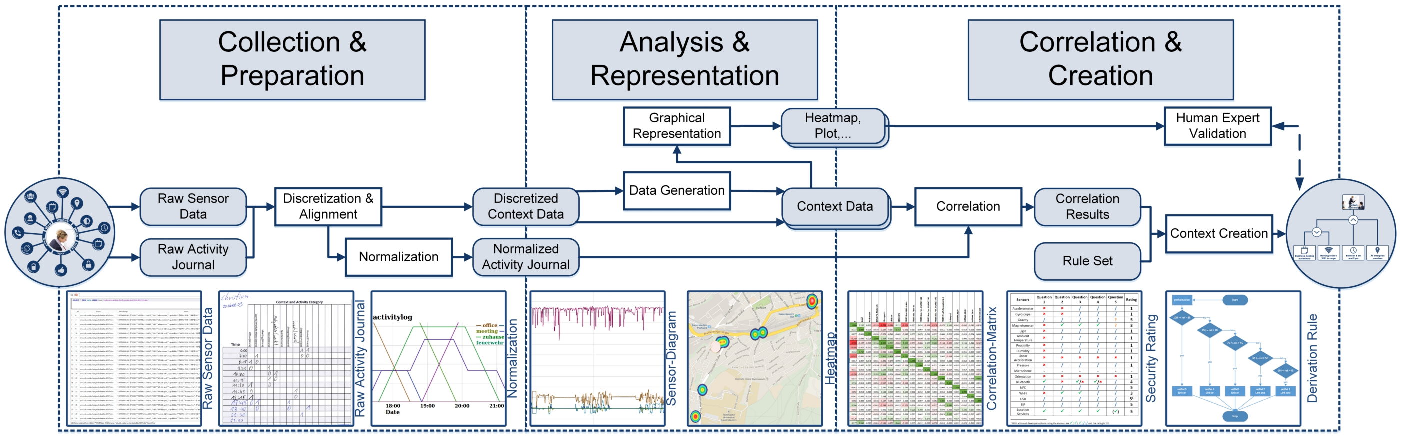


Figure 7. Detailed Process for Determining Context Descriptions

The entire process comprises three phases, depicted in Figure 7:

- Collection & Preparation,
- Analysis & Representation, and
- Correlation & Creation.

1) Collection & Preparation: In the first phase, we manually collect user activities/device situations and use mobile devices to automatically collect related contextual information (sensor data). The two data streams are fed into the process for further processing. We are using discrete time steps in the subsequent phase and therefore, we have to discretize the data collected. We have different strategies to transform the information, which depend on the nature of the given data. In addition, we consolidate the reported user activities and situations regarding their semantics (as users may use different verbalisms for reporting their activities and situations).

2) Analysis & Representation: The second phase has two objectives. First, the discretized context data can be analyzed to derive new information from it. For example, changes in battery level over time can be derived from absolute battery levels at specific points in time. Second, we create graphical representations of the data sets. For example, geolocation over time can be visualized via heatmaps. The graphical representations is used to allow human experts to validate context descriptions in the end.

3) Correlation & Creation: In the third phase, context descriptions are derived. Towards this end, the normalized activity journals are first correlated with contextual data. We use different statistical methods (see [8]) to correlate sensor data with activities/situations and use the results (e.g., a correlation matrix) to determine the relevance of a sensor for the characterization of an activity or situation. Derivation rules are then applied, producing context descriptions that correspond to user activities.

The evaluation result of a context is obtained by evaluating the tree structure of the context description. The logical operators have their standard meaning.

The calculation of the relevance and security rating for a context is as follows:

- **AND/OR**-relation: All fulfilled quality attributes of the elements in an AND/OR group affect the overall relevance or security rating. They are summed up to the denominator. The quality attributes of all evaluators that are actually fulfilled in the system under evaluation are summed up to the numerator.
- **NOT**-relation: The quality attribute of the element is propagated to the parent node, if the subordinated expression is false.

The quality attributes ensure that the fulfillment of those evaluators with highest relevance or security rating has the strongest impact on the overall result. For example, let us assume a context description c with three evaluators e_1, e_2, e_3 linked with a logical or: $c = e_1 \lor e_2 \lor e_3$. Assume further that evaluator $e_1(true,\ sec = 1,\ rel = 1)$ is fulfilled and has a relevance and security rating of one, and that $e_2(false,\ sec = 3,\ rel = 4)$ and $e_3(false,\ sec = 4,\ rel = 5)$ are not fulfilled and have a relevance rating of four and five, respectively. Then, the overall result of the context is fulfilled, but the relevance rating is only $1/10 = 0.1$ and the security rating is $1/8 = 0.125$. The security policy specification bears responsibility for defining suitable thresholds for the security and relevance ratings that are sufficient to trigger a change of the security settings. Furthermore, the decision strongly depends on whether to tighten or to ease security restrictions.

IV. Application Scenario

To demonstrate how such an approach can be used in an industrial setting, we applied it in cooperation with a large German company that administrates mobile devices via the mobile device management (MDM) solution *MobileIron*®. Via MobileIron®, they adjust security settings (e.g., to impose password restrictions or storage encryption, to install or revoke certificates for virtual private networks, or to disable camera or microphone), and perform actions such as sending messages to the user or wiping the device. However, these settings and actions are rather static and cannot be adapted according to the current operational state of the device or the user activity. In this setting, context-awareness can provide more flexibility. For example, camera and microphone usage can be prevented within company premises, but allowed elsewhere. However, when used for security purposes, context detection has to be accurate and reliable in order to comply with company regulations.

In our application, we analyzed the security demands of the company and identified the needed flexibility. We created appropriate context descriptions and connected our context detection with the MDM solution of the company. Unfortunately, we were not allowed to run the evaluation with productive users and had to evaluate it with researches from our institute.

A. Setup & Execution

Security policies in MobileIron® control security behavior such as password restrictions of the mobile device (cf. Table I). *Lockdown policies* limit the use of the mobile device such as disabling Bluetooth, camera, or microphone (cf. Table II). We specified two security policies ($security1$ and $security2$) and three lockdown policies ($lockdown1$, $lockdown2$, and $lockdown3$), which are briefly described in the following.

TABLE I. Security Policies

	security1	security2
Maximum Inactivity Timeout:	30 min	2 times
Maximum Number of Failed Attempts:	3 min	5 times

TABLE II. Lockdown Policies

	lockdown1	lockdown2	lockdown3
Bluetooth:	Disable	Enable (Audio only)	Enable
Camera:	Disable	Enable	Enable
Microphone:	Disable	Enable	Enable
NFC:	Disable	Enable	Enable
Screen Capture:	Disable	Enable	Enable
Lockscreen Widgets:	Disable	Enable	Enable
USB Debug:	Enable	Disable	Enable

In Table I, security policy $security1$ has a higher priority than security policy $security2$. Hence, if both policies are activated, security1 would be used and the maximum inactivity time would be 30 minutes. In Table II, the lockdown policy $lockdown1$ has a higher priority than $lockdown2$, which in turn has a higher priority than $lockdown3$.

MobileIron® allows the definition of labels and the assignment to security and lockdown polices. In our case, we defined four labels, namely $default\_label$, $work1\_label$ where security is tightened, $work2\_label$ where security is eased and $home\_label$. These labels have been assigned to our policies as depicted in Figure 8.

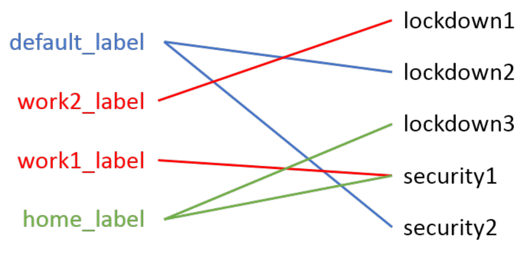


Figure 8. Mapping between Labels and Policies.

In the described setting, two contexts are of relevance: “Home” and “Work”. Both contexts are modeled by using wireless, location, calendar, and time evaluators. Figure 9 illustrates the context tree for the “Work” context c_1, including all assignments for the security and relevance ratings. As the security rating has a global scope for each evaluator type, the value does not change within the same type of evaluator. In contrast, the relevance rating changes, depending on the results from the statistical calculation.

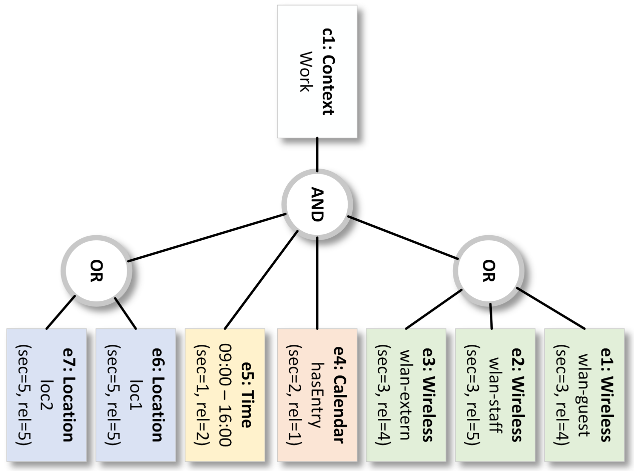


Figure 9. Context Tree Example for Context “Work”.

For example, the wireless network *wlan-staff* has the highest statistical significance (correlation result), followed by *wlan-guest* and *wlan-extern*. This is reflected in the final relevance ratings of the wireless evaluators. *wlan-staff* is the employer’s wireless network and has the highest relevance. The calendar seems to be a relevant context factor, but as users usually do not schedule their entire working day in the calendar, the calendar evaluator has the lowest relevance. Similar behavior holds for the time evaluator. The company has flexible working hours, but the core working hours are between 09:00am to 4:00pm. Hence, users arrive earlier or stay longer at work, to reach their daily working time. Nevertheless, the time evaluator is more relevant than the calendar evaluator.

The company has defined several policies assigned to the “Work” context, on which we focus in the following. On the one hand, there are policies easing security restrictions of the mobile device at work (in favor of usability). For example, the company increases the display timeout at work to thirty minutes for usability reasons ($work1\_label \rightarrow security1$). On the other hand, there are policies tightening the security at work. For example, the company prohibits the usage of camera, microphone, etc. at work to meet organizational policies ($work2\_label \rightarrow lockdown1$). The idea is now to define appropriate thresholds to reflect company needs.

To tackle this, we calculated the following cases:

- Context c_1 is $true$ with highest relevance $\rightarrow 1.00$
- Context c_1 is $true$ with lowest relevance e_1, e_4, e_5, e_6 are $true \rightarrow 0.46$
- Context c_1 is $false$ with highest relevance $e_1, e_2, e_3, e_5, e_6, e_7$ are $true \rightarrow 0.96$
- Context c_1 is $false$ with lowest relevance $\rightarrow 0.00$

Hence, the relevance range for c_1 $is\ true$ is between 0.46 and 1.00, and for c_1 $is\ false$ is between 0.00 and 0.96. Analogously, we calculated the security rating for c_1. Figure 10 shows our policy state chart and the state change criteria to activate and deactivate the policies. To change the states by using the relevance and security rating, as well as the fulfillment of the context, allows us to model hysteresis behavior. For example, changing from the state $work1\_label$ (inactive) to $work1\_label$ (active) is harder than changing from the state $work1\_label$ (active) to $work1\_label$ (inactive).

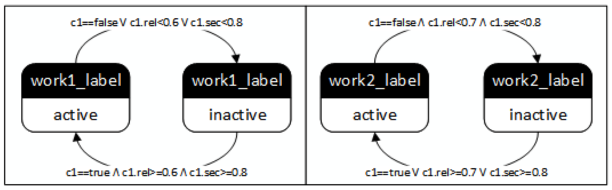

c1==false V c1.rel<0.6 V c1.sec<0.8
work1_label
active
work1_label
inactive
c1==true Λ c1.rel>=0.6 Λ c1.sec>=0.8
c1==false Λ c1.rel<0.7 Λ c1.sec<0.8
work2_label
active
work2_label
inactive
c1==true V c1.rel>=0.7 V c1.sec>=0.8

Figure 10. Policy Statechart.

For instance, assume that $work1\_label$ is inactive, the evaluators e_1, e_4, e_5, e_6 are true, and the evaluators e_2, e_3, e_7 are false. This results in context c_1 to be true. However, the relevance is only 0.46 and the security is only 0.50, which would prevent the change from inactive to active. Hence, we need at least one more evaluator changing its state to true for reaching the relevance threshold and even one more for reaching the security threshold. Vice versa, let us suppose that $work1\_label$ is active and the evaluators have the same state as before. Although c_1 is still fulfilled, we would make the change as the ratings are below the defined thresholds of 0.60 (relevance) and 0.80 (security).

B. Lessons Learned

The practical application was performed by two of our internal researchers, as they had to manually observe the mobile devices. For our case study, we used a Samsung Google Nexus 10 running Android 5.1 and a Samsung Galaxy Tab (SM-P600) running Android 4.4.2. Both devices were added to the MDM solution and had to adhere to the company-specific security policies. We made several observations in our practical application, which we describe next.

Some sensors have uncertainties and inaccuracies in their measurements. In our scenario these are the location and the wireless sensors. We configured the wireless sensor to scan for specific wireless networks every five minutes and prevented the user from disabling by setting the *keepEnabled* flag. However, the sensor occasionally misses some wireless networks although the networks are available. If we measure for a specific wireless network ten times, the sensor will miss this specific network one to two times. Hence, we have a failure rate of 10 to 20 percent in our measurements. Let us assume our context "work" c_1 is fulfilled and $work1\_label$ is active as well as $work2\_label$ is active. The affected wireless evaluators are e_1, e_2, e_3. All other evaluators are assumed to be fulfilled, i.e., working correctly. As they are in OR-relation (cf. Figure 9), all three have to provide a wrong measurement to yield an overall evaluation result of c_1 that is wrong, which did not happen during our evaluation period. Evaluator e_2 has the highest impact on the relevance and security ratings. If the evaluation of e_2 fails, the relevance rating is at 0.81 and the security rating is at 0.86. Both ratings are above the specified thresholds to trigger a state change for the labels $work1\_label$ and $work2\_label$. The failure of an additional wireless evaluator, for instance e_1 or e_3, puts the ratings to 0.65 (relevance) and 0.73 (security), which are below the thresholds. Such ratings result in a change of $work1\_label$ from active to inactive, which is uncritical as it tightens our security settings. Regarding $work2\_label$, we will stay in the active state, as the overall context c_1 is still true, which is also uncritical. To trigger a state change, we would need all three evaluators to fail, which did not happen during our evaluation period, as already mentioned.

Regarding the location evaluation, we observed that we have some uncertainties in the location evaluation of e_6 and e_7 when people are entering the specified locations. Such location changes usually happened in the morning, when people arrived at work, and after noon, when people came back from lunch outside the company. The reason for detection failures is the inaccurate location fix after a location change. The Android location services return a coarse grained location, which is outside our specified locations for the evaluators. Let us assume our context "work" c_1 is not fulfilled and $work1\_label$ is inactive as well as $work2\_label$ is inactive. The affected wireless evaluators are e_6, e_7. All other evaluators are assumed to be fulfilled, i.e., to work correctly. To trigger a state change, both evaluators have to be evaluated to true. Regarding $work1\_label$, it is uncritical as we are remaining in the high security settings; however, $work2\_label$ is critical. As we configured to receive a location fix latest every five minutes and after location changes greater than fifteen meters, we may stay five to ten minutes in a wrong state. However, as the Android location services pushes new information after the location fix, we observed to stay less than five minutes in the wrong state.

We modeled all evaluator groups in AND-relation, which was a bad decision regarding the time and calendar evaluators. As the working hours was given between 09:00am and 04:00pm, e_5 was also configured to be true in the specified interval. However, people usually do not completely stick to these working hours. We observed that $work1\_label$ stayed too long in state inactive (starting working before 09:00am) or changed from active to inactive too early (working longer than 04:00pm). Similar observations were made for the calendar evaluator e_4. We learned to model evaluators with lower relevance in an OR-relation rather than in an AND-relation. However, we have to gain more experience to make a final decision.

Overall, we conclude that our approach is feasible to adapt security settings provided by the MDM solution in our given scenario. Future work has to analyze the performance, focusing on the latency between the context detection and the effective enforcement of the security settings on the mobile device.

V. Experimental Evaluation

In our practical application, we evaluated our approach in a company setting. However, we run tests with two researchers in our premises and were not allowed to monitor the behavior in the company. In addition, our researchers did not use the mobile devices for their daily work. Hence, we did not evaluate the correctness of the context detection.

We started another experiment to evaluate precision and recall of our context detection in a real world setting. The following steps have been performed:

- **Preparation 1 - Data Collection:** Automated collection of contextual data and manual reporting of user activities and situations. This data is required for the context description generation.
- **Preparation 2 - Context Description Generation:** Derivation of context descriptions by using the process depicted in Figure 7. The derived context descriptions are used for the real world context detection.
- **Execution - Real World Context Detection:** Detection of context by using the generated context descriptions to adapt mobile device behavior (for convenient and security reasons) in a real world application.
- **Analysis & Discussion:** Analysis of the reported context detections with respect to precision and recall.

A. Preparation 1 - Data Collection

We first performed some data collection experiment before our actual evaluation experiment. In our first experiment, seven voluntary participants collected their contextual real-life data over a period of four weeks. The collection has two main parts: Subjects manually report their activities and situations in a context journal. Concurrently, mobile devices automatically collect contextual information from different sources; for example, physical sources such as built-in sensors (e.g., accelerometer, wireless networks) and virtual sources such as calendar entries.

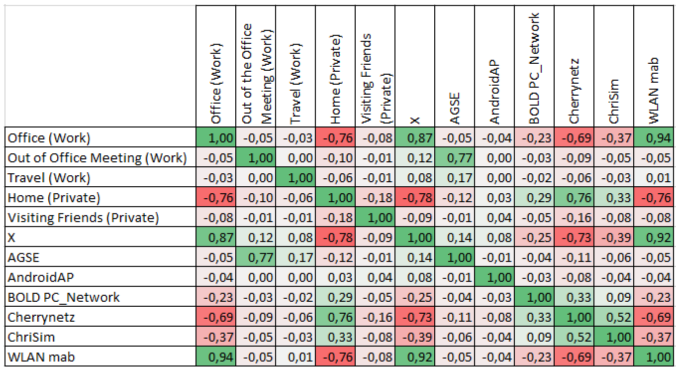
For data collection, we use a mobile application, the funf framework [9]. The funf team implemented the application for sensing and processing contextual information on smart mobile devices. We use the "funf Journal" app from Google's PlayStore [10]. It allows a flexible sensing, processing, and storing of contextual information. We can configure the mobile application to collect data from different context information sources. The mobile app allows flexible scheduling based on time constraints and configurable triggers. Funf supports 38 different context information sources for data collection, so-called probes. Funf divides them into the following categories: device, device interaction, environment, motion, positioning, and social. After configuring and starting the funf application, it autonomously collects data as background process and stores the results in a SQLite database on the mobile device. The user can export the database to the local file system or upload the data to a server. Overall, we configured fourteen different sensors that collected about 110 million data entries in four weeks.

Besides raw context data, we also need information about the current activity/situation, in which the subject acts during the measurements for determining the relation between raw sensor data and user activities/device situations. Hence, our participants manually report their perceived context during the data collection phase in a context journal. The participants reported their activities paper-based or using the mobile application "Gleeo Time Tracker" [11].

During the automated context data collection, the participants are also reporting their activities. We tell the participants to check whether there are still active activities when they report a new activity. This is only important for the reporting on paper as the Gleeo Time Tracker app automatically stops running tasks when the user starts another task. Overall, the participants manually reported 65 different activities and situations during the measurement. Most of the activities occurred only a few times and could not be used in our process, as we are using statistical methods to calculate the correlation between contextual information and activities/situations. The three most relevant contexts for our experiment and further evaluation are "home", "work" and "sleeping".

B. Preparation 2 - Context Description Generation

In this step, the contextual and the journal data are fed to the derivation process to produce context descriptions. To determine the correlation, we choose binary correlation as statistical method. The method produces a correlation matrix considering all variables. Figure 11 shows binary correlations of reported activities followed by wireless networks. The values range from -1, denoting maximal anti-correlation, to +1, denoting maximal correlation. For instance, we take the first line with the activity "Office (Work)", which has a high correlation with "X" (0.87) and "WLAN mab" (0.94). Contrary, "Office (Work)" has a high anti-correlation with "Home (Private)" (-0.76) and "Cherrynetz" (-0.69). The anti-correlation between the two activities "Home (Private)" and "Office (Work)" is an obvious result.



	Office (Work)	Out of the Office Meeting (Work)	Travel (Work)	Home (Private)	Visiting Friends (Private)	X	AGSE	AndroidAP	BOLD PC_Network	Cherrynetz	ChriSim	WLAN mab
Office (Work)	1,00	-0,05	-0,03	-0,76	-0,08	0,87	-0,05	-0,04	-0,23	-0,69	-0,37	0,94
Out of Office Meeting (Work)	-0,05	1,00	0,00	-0,10	-0,01	0,12	0,77	0,00	-0,03	-0,09	-0,05	-0,05
Travel (Work)	-0,03	0,00	1,00	-0,06	-0,01	0,08	0,17	0,00	-0,02	-0,06	-0,03	0,01
Home (Private)	-0,76	-0,10	-0,06	1,00	-0,18	-0,78	-0,12	0,03	0,29	0,76	0,33	-0,76
Visiting Friends (Private)	-0,08	-0,01	-0,01	-0,18	1,00	-0,09	-0,01	0,04	-0,05	-0,16	-0,08	-0,08
X	0,87	0,12	0,08	-0,78	-0,09	1,00	0,14	0,08	-0,25	-0,73	-0,39	0,92
AGSE	-0,05	0,77	0,17	-0,12	-0,01	0,14	1,00	-0,01	-0,04	-0,11	-0,06	-0,05
AndroidAP	-0,04	0,00	0,00	0,03	0,04	0,08	-0,01	1,00	-0,03	-0,08	-0,04	-0,04
BOLD PC_Network	-0,23	-0,03	-0,02	0,29	-0,05	-0,25	-0,04	-0,03	1,00	0,33	0,09	-0,23
Cherrynetz	-0,69	-0,09	-0,06	0,76	-0,16	-0,73	-0,11	-0,08	0,33	1,00	0,52	-0,69
ChriSim	-0,37	-0,05	-0,03	0,33	-0,08	-0,39	-0,06	-0,04	0,09	0,52	1,00	-0,37
WLAN mab	0,94	-0,05	0,01	-0,76	-0,08	0,92	-0,05	-0,04	-0,23	-0,69	-0,37	1,00

Figure 11. Correlation Matrix (Binary Correlation)

The correlation matrix is used to generate operative context descriptions for the three relevant contexts: "home", "work" and "sleeping". In general, the generation process can optimize the produced context descriptions in terms of precision and recall. Precision is the relation of true positives to true positives and false positives. It denotes that if our context test detects a specific context, it will be correct or not. Contrary, recall is the relation of true positives to true positives and false negatives. It denotes that if a context is happening in the real world, our test will detect it. For our evaluation, we decided to choose context descriptions optimized in terms of precision for the transition $0 \rightarrow 1$. Hence, the precision should be high for detecting the entering into a specific context.

C. Execution - Real World Context Detection

In the actual experiment four voluntaries participated (out of the seven voluntaries for the first experiment). For these four participants, we take precision-optimized context descriptions to detect "home", "work", and "sleeping" (based on the data from the collection experiment). The voluntaries used the following private mobile devices to participate in the evaluation:

- Participant P1, Nexus 5 running Cyanogenmod 12.1
- Participant P2, Nexus 5 running stock ROM 4.4.4
- Participant P3, Nexus 4 running Cyanogenmod 12.1
- Participant P4, HTC One (M7) running Cyanogenmod 11

We equipped their private mobile devices with the Integrated Distributed Data Usage Control Enforcement (IND^{2}UCE) framework for Android [12] where the following features were used in our evaluation:

- **Device Administration:** The framework acts as device administrator for Android [13]. Hence, it supports device administration features such as changing security related settings, wiping, or locking the device, and disabling the camera.
- **Volume Control:** The framework supports controlling settings related to the volume control of the mobile device (i.e., setting the ringtone volume).
- **Captive Portal Login:** One extension addresses the automated login to a captive portal for granting access to the internet (without having to manually login via a webpage)
- **Network Settings:** We are able to control network settings with the framework. A simple example is the activation or deactivation of Bluetooth or Wifi.

We created policies for the IND^{2}UCE framework to react on context changes. Hence, we created a security policy for every possible context transition:

- Context was not fulfilled and is now fulfilled: $0 \rightarrow 1$-Transition
- Context was fulfilled and is now not fulfilled: $1 \rightarrow 0$-Transition

Such a context transition is modeled as depicted in Figure 12. The condition is checked every five minutes, as configured in the *timestep*-tag. In the condition, we resolve the actual context by using a so-called Policy Information Point (PIP) with the name "context". As parameter, we add the context id, which is "P1_home" in our case for participant P1. We link this request with a *before*-operator by using a logical *and*. Within the *before*-operator, we have a negated PIP request with the same parameters as our first one. To sum it up, if the first PIP request is now true and the second PIP request has been false in the timestep before, the condition will be true. Hence, we can phrase it as follows: We had a context change for the context "P1_home" from being not fulfilled to being fulfilled within the last five minutes.

We had two meetings with the participants to elicit convenient and security functions, they wanted to have depending on their current context. The focus was set to the three contexts: "home", "work", and "sleeping". Hence, every participant stated wanted behavior when, for instance, she is coming home or leaving home. Here are some examples we elicited:

- "When I am leaving Fraunhofer IESE [work], I would like to set my ringtone volume to 100%."
- "When I am at Fraunhofer IESE [work], I would like to set my ringtone volume to 20%."
- "When I am at home, I would like to activate the Wifi."

```
<timestep amount="5" unit="MINUTES" />
<condition>
 <and>
  <pip:boolean name="context" default="true">
   <param:string name="value" value="P1_home" />
  </pip:boolean>
  <before amount="1" unit="TIMESTEPS">
   <not>
    <pip:boolean name="context" default="true">
     <param:string name="value" value="P1_home" />
    </pip:boolean>
   </not>
  </before>
 </and>
</condition>
```

Figure 12. Transition Example for $0 \rightarrow 1$-Transition for Context "P1_home"

- "When I am not at home and not at work, I would like to deactivate the Wifi."
- "When I am at home, in my bedroom, and the time is between 10pm - 7am, then activate FlightMode."
- "When I am leaving my home, I would like to set my ringtone volume to 100%."

We elicited 29 rules to be applied based on the context detection. However, we are only able to fulfill the requirements of 20 rules with the IND^{2}UCE framework.

```
<detectiveMechanism name="home_0->1">
  <description>
  Context change from false to true
  </description>
  <timestep amount="5" unit="MINUTES" />
  <condition>
   <and>
    <pip:boolean name="context" default="true">
     <param:string name="value" value="P1_home" />
    </pip:boolean>
    <before amount="1" unit="TIMESTEPS">
     <not>
      <pip:boolean name="context" default="true">
       <param:string name="value" value="P1_home" />
      </pip:boolean>
     </not>
    </before>
   </and>
  </condition>
  <!-- setRingToneVolume to 50% -->
  <executeAction name="urn:action:local:volume">
   <param:int name="level" value="3"/>
  </executeAction>
  <!-- setDisplayTimeout to 30min -->
  <executeAction name="urn:action:local:SecuritySettings">
   <param:int name="display" value="30"/>
  </executeAction>
  <!-- feedback for context change -->
  <executeAction name="urn:action:local:feedback">
   <param:string name="mode" value="context" />
   <param:string name="value" value="P1_home" />
   <param:string name="flag" value="1" />
  </executeAction>
 </detectiveMechanism>
```

Figure 13. Policy Mechanism Example for $0 \rightarrow 1$-Transition for Context "P1_home"

We created for every participant appropriate rules, which we call policies in the IND^{2}UCE framework. Figure 13 illustrates a policy for the handling of the $0 \rightarrow 1$-Transition for context "P1_home". In other words, it handles what will happen when the participant is coming home. There are four execute actions to be performed by the IND^{2}UCE framework: First, the framework will set the ringtone volume to about 50%, which is done by the execute action "volume". Second, IND^{2}UCE sets the display timeout to 30 minutes by performing the execute action "SecuritySettings". Third, we will trigger our feedback app. When the feedback app is triggered in mode "context", it displays a sticky notification to the user and possibilities for the user to confirm or reject (cf. Figure 14):

Verify context detection 15:25
"*atWork*" is now active
Yes / No

A "yes"-button means, the context detection was right and the user confirms the correctness. By clicking on the "yes"-button, we count the context detection as correct. A "no"-button means, the context detection was wrong and the user is forwarded to a list of already recorded other contexts. If the context is not in the list, the user can add a new context. We count the context detection as wrong and add a new entry to the table with the context entered by the user. Doing so, we are capable to count all true positive and false positive context detections and can easily calculate the precision of the context detection.

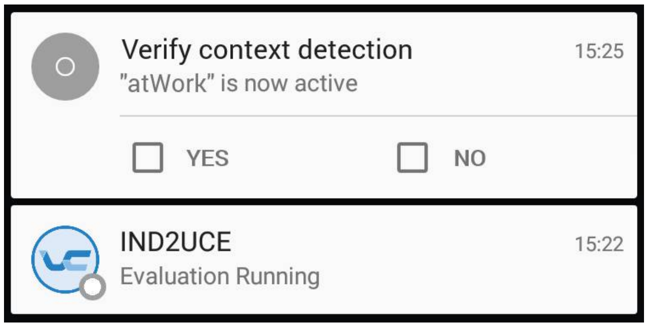

Verify context detection
15:25
"atWork" is now active
YES
NO
IND2UCE
15:22
Evaluation Running

Figure 14. Context Check PXP: Notification

If the user misses to process the notification, we store all information in a list of unanswered context detection entries (see Figure 15). The user has the possibility to answer the questions later. By clicking on the "yes"-button or "no"-button, the app behaves as in the notification interaction.

We conducted the experiment over a period of two weeks, in which the participants had their usual working times (no vacation, no business trips, etc.). Unfortunately, we experienced technical difficulties for one participant and had to continue the experiment without him. For the three other participants, everything worked as expected.

D. Analysis & Discussion

The participants reported 243 context detections in two weeks, which results in about six context detections per day (in average). This is a plausible value, as it can be seen in the following reported feedback schedule of one of the participants (counting eight context detections):

However, the feedback occurrence distribution is 96 for P1, 44 for P2, and 103 for P3. P1 and P3 are nearly equal, P2 has much less context changes detections. There are two explanations for this behavior: First, P2 usually stays at work for having lunch and is not leaving the institute. Second, there are less changes for the context "home" in contrast to the other two participants.

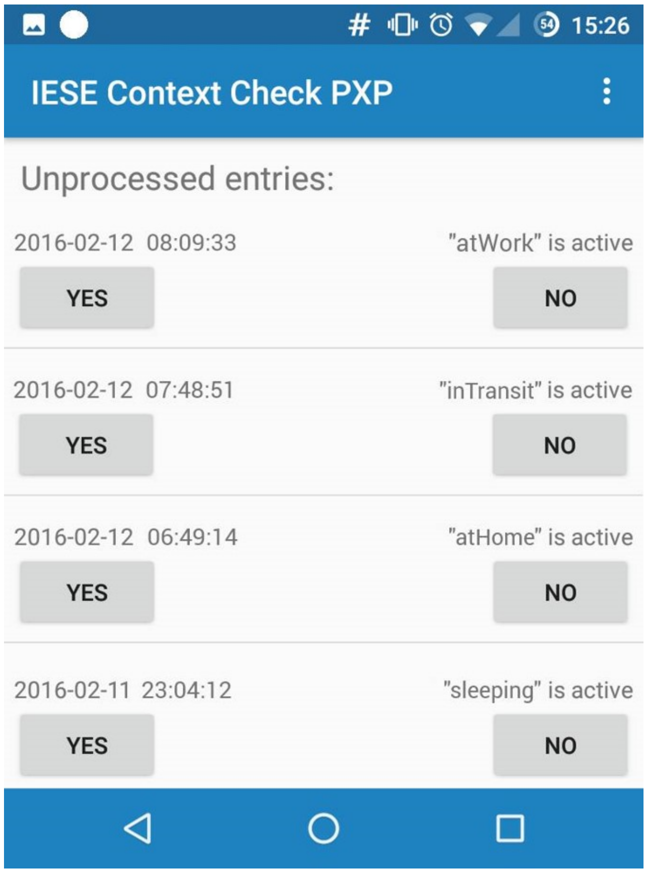

15:26
IESE Context Check PXP
Unprocessed entries:
2016-02-12 08:09:33 "atWork" is active
YES NO
2016-02-12 07:48:51 "inTransit" is active
YES NO
2016-02-12 06:49:14 "atHome" is active
YES NO
2016-02-11 23:04:12 "sleeping" is active
YES NO

Figure 15. Context Check PXP: Unprocessed Entries

TABLE III. Excerpt of Feedback Result

No.	Time	Context	Transition	Description
1	15.10.2015 07:08	sleeping	$1 \rightarrow 0$	waking up
2	15.10.2015 08:06	home	$1 \rightarrow 0$	leaving home
3	15.10.2015 08:38	work	$0 \rightarrow 1$	entering work
4	15.10.2015 11:58	work	$1 \rightarrow 0$	leaving work (e.g., lunch)
5	15.10.2015 12:36	work	$0 \rightarrow 1$	entering work
6	15.10.2015 16:41	work	$1 \rightarrow 0$	leaving work
7	15.10.2015 17:47	home	$0 \rightarrow 1$	entering home
8	15.10.2015 22:57	sleeping	$0 \rightarrow 1$	sleeping

Table IV illustrates the overall result of the context detection for every participant. The first identifies the participant. The second and third column represents whether the context detection was reported as correct or wrong. The fourth and last column indicates the user input, if the context detection was wrong. As shown in the table, participant P1 and P2 always corrected the context detection, participant P3 missed it two times.

TABLE IV. Context Detection Results

Participant	Context Change Detected		
	Correct	Wrong	User Input
P1	53 (55.21%)	43 (44.79%)	43
P2	32 (72.73%)	12 (27.27%)	12
P3	76 (72.38%)	29 (27.62%)	29

The correctness of the context detection is between 55.21% and 72.73%. The result is not very convincing for being used in security related settings. However, the figures represent the overall correctness of the context detection. As we created precision-optimized context descriptions for the $0 \rightarrow 1$-transitions, we have to split up the data. The result is presented in Table V.

TABLE V. Context Detection Result (split)

Part.	Context Change Detected ($0 \rightarrow 1$)			Context Change Detected ($1 \rightarrow 0$)		
	Correct	Wrong	Sum	Correct	Wrong	Sum
P1	44 (95.65%)	2 (4.35%)	46	9 (18.00%)	41 (82.00%)	50
P2	18 (100.0%)	0 (0.0%)	18	14 (53.85%)	12 (46.15%)	26
P3	51 (98.15%)	1 (1.85%)	54	23 (45.10%)	28 (54.90%)	51

Now, we get a much better result regarding context detection precision for the $0 \rightarrow 1$-transitions. All values are above 95%, which fulfills our expectations. Moreover, we made no error for participant P2 and reached 100%. From our expectations, this is a good result and such context descriptions could be used to ease security mechanisms.

However, the precision for the context detection for the $1 \rightarrow 0$-transitions ranges from 18% to nearly 55%. In other words, the context detection would be wrong for every second detection in best case. This result cannot be used to adapt security mechanisms with high precision. We could improve the precision by either adapting the parameters for the hysteresis behavior (cf. Section IV) or using precision-optimized context descriptions also for the $1 \rightarrow 0$-transitions.

Obviously, there is a trade-off between these two kinds of context changes and we managed to optimize our context detection only in one direction (i.e., $0 \rightarrow 1$-transitions). Hence, we achieved a high precision for easing security mechanisms, but allow lower precision for tightening security mechanisms. In addition, we have to achieve a higher recall rather than having a high precision for tightening security mechanisms. Regarding the recall of our approach, we cannot provide reliable figures for this calculation as we have not monitored the real world behavior. Hence, we do not know how often or long the user was already in the specific context, but the context detection did not recognize.

VI. Related Work

This section provides an overview of the state of the art in context-awareness and context-aware computing. More specifically, the term *context* and its early definitions are introduced and different context modeling approaches are described.

A. *Definitions of the term "Context"*

The notion of *context* emerged over time, the earliest definitions in our sense originate in the early nineties. All of them share similarities, but they also show differences. We will present the most prevalent and influential definitions.

In 1994, an early definition was provided by Schilit et al. [3], stating that a context is characterized by three aspects: where you are, whom you are with, and what resources are nearby. Therefore, Schilit et al. infer that context-aware systems have to depend on the location of use, nearby people, hosts, and accessible devices, as well as on changes over time. Although they state that context is more than just location, location is apparently a very important information source for context-aware systems. Interestingly, although smart mobile devices did not nearly have the same capabilities in 1994 as today's devices have (especially the plethora of sensors), the authors already named today's sensors (e.g., light intensity, network connectivity) as additional context sources.

In 1997, Brown and Bovey [14] describe context similar to Schilit et al. but include temporal attributes, such as time of day, season, and temperature, as additional contextual information sources. In addition, the authors propose to enrich context by using additional (user-provided) information to obtain more valuable information for their application.

Hull et al. [15] describe context as "many aspects of a user's situation", such as "user identity, location, companions, vital signs, air quality, and network availability". Franklin and Flachsbart [16] focus on intelligent environments observing their users. They state that context-aware computing should consider the observed situation of the user. A similar description can be found in [17]. Ryan et al. [18] state that context should include location as well as states of external and internal sensors of the computer itself. Hence, they also consider virtual context sources such as the state of the software running on the device. Pascoe [19] also considers virtual context sources, but describes them as the states of the application and its environment rather than states of the computer itself. Pascoe et al. [20] reveal the more rich and complex nature of context and that context can be complex. Furthermore, in accordance with other publications, they state that context is more than just location.

In 1999, Abowd et al. [21] define context as any information that is used to characterize the situation of an entity. As mentioned in other publications, context is seen as additional information or as an attribute of an entity. They also state that context information has to be categorized in different context types, which makes it easier for context-aware computing. They introduce four primary context types for characterizing an entity's situation: location, identity, time, and activity.

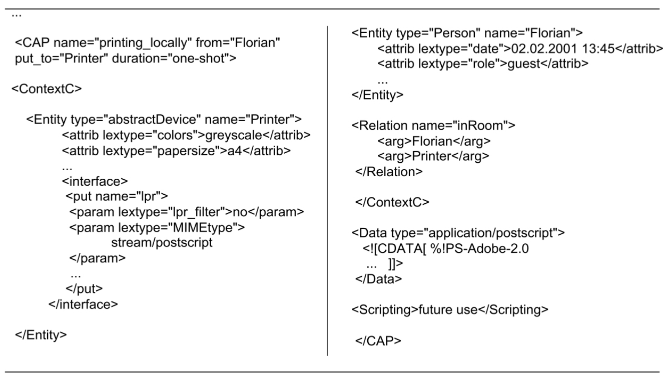
Hofer et al. [6] partition context information regarding its origin and differentiate between physical, virtual, and logical context information. Physical context information, such as location, acceleration or light intensity, can directly be measured by sensors. Such physical measures are described as low-level context sources that are continuously updated. Virtual information stems from user data or internal system data. The latter context category, logical context information, is obtained by combining physical and virtual context sources according to some abstract logical rules.

B. *Modeling Context Information*

Context-aware systems strongly rely on the quality of the context information, which is usually represented in a context model. The modeling and provision of context information is very important to fulfill the desired task. In this work, context awareness aims at the enforcement and adaptation of flexible security policies on the mobile device and its applications. Different approaches for modeling context information have been suggested. In [22] and [23], the authors survey the most relevant approaches and classify them into five categories:

Key-Value Models are the simplest model for structuring context information. As such models provide no structuring of information, they are easy to manage. They are often used, although they provide only limited support for more sophisticated modeling [22][23]. Key-Value models allow easy querying by simple algorithms matching the key value pairs. The querying can be enriched by Boolean operators or wild-cards for the matching algorithm.

Markup Scheme Models use a hierarchical structure of markup tags containing attributes and their values. A well-known example is the eXtensible Markup Language (XML). A markup scheme has been proposed, for instance, in [24]. In contrast to key-value models, markup scheme models provide a mechanism for structuring context information. However, querying such models becomes more complex than in the simple key-value model, but is essentially done similarly by matching the values of the markup tags or their attributes. In [25], Samulowitz et al. define "Context-Aware Packets (CAPs)" for storing context information. CAPs are organized in "context constraints, scripting, and data", where the context constraints part contains the modeled context information. This sub part is in turn subdivided into "abstract entities, relations, and events". Figure 16 illustrates a CAP example taken from [25]. The given CAP describes two entities (i.e., Printer and Florian) and a relation "inRoom" between the two entities. The relation describes the context "ContextC".



```
...

 <CAP name="printing_locally" from="Florian"
  put_to="Printer" duration="one-shot">

<ContextC>

   <Entity type="abstractDevice" name="Printer">
          <attrib lextype="colors">greyscale</attrib>
          <attrib lextype="papersize">a4</attrib>
          ...
          <interface>
           <put name="lpr">
            <param lextype="lpr_filter">no</param>
            <param lextype="MIMEtype">
                    stream/postscript
            </param>
            ...
           </put>
        </interface>

 </Entity>

<Entity type="Person" name="Florian">
      <attrib lextype="date">02.02.2001 13:45</attrib>
      <attrib lextype="role">guest</attrib>
      ...
</Entity>

<Relation name="inRoom">
      <arg>Florian</arg>
      <arg>Printer</arg>
 </Relation>

 </ContextC>

<Data type="application/postscript">
  <![CDATA[ %!PS-Adobe-2.0
   ...  ]]>
 </Data>

<Scripting>future use</Scripting>

 </CAP>
```

Figure 16. Context-Aware Packets Example from [25]

Graphical Models can strongly vary in their representation. The best-known representative is probably the Unified Modeling Language (UML), which is also suitable for modeling context, as shown in [26] or [27]. Such models are easy to understand for human beings, but often lack formality.

Henricksen et al. [28] present a context extension for the Object-Role Modeling (ORM) approach, which describes context as collection of facts (as shown in Figure 17).

The model can directly be used to derive an entity relationship model as a basic structure for relational databases. An interesting aspect of their model is the differentiation between static context information (i.e., facts that remain unchanged as long as the entities they describe persist) and dynamic context information. They distinguish between contextual information that can be treated as property or constant attribute and changing contextual information such as location.

Object Oriented Models provide their information as a collection of objects that contain context information. Such models can employ all object-oriented modeling techniques

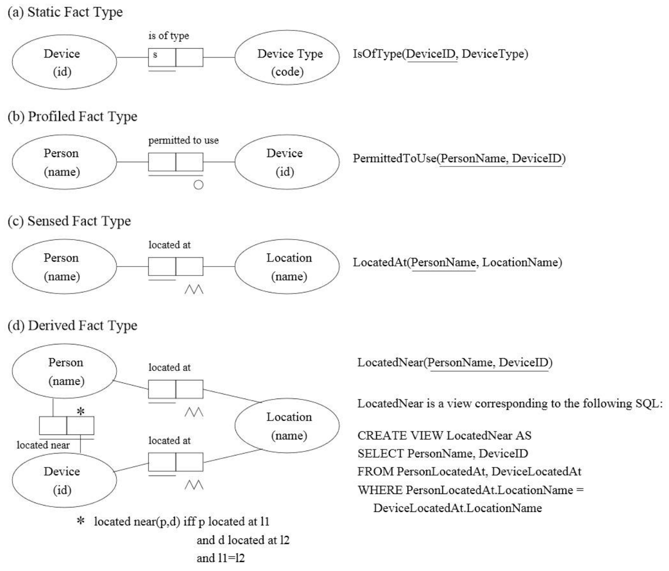

(a) Static Fact Type
Device (id)
is of type
s
Device Type (code)
IsOfType(DeviceID, DeviceType)
(b) Profiled Fact Type
Person (name)
permitted to use
Device (id)
PermittedToUse(PersonName, DeviceID)
(c) Sensed Fact Type
Person (name)
located at
Location (name)
LocatedAt(PersonName, LocationName)
(d) Derived Fact Type
Person (name)
located at
located near
Device (id)
located at
Location (name)
LocatedNear(PersonName, DeviceID)
LocatedNear is a view corresponding to the following SQL:
CREATE VIEW LocatedNear AS
SELECT PersonName, DeviceID
FROM PersonLocatedAt, DeviceLocatedAt
WHERE PersonLocatedAt.LocationName =
DeviceLocatedAt.LocationName
* located near(p,d) iff p located at l1
and d located at l2
and l1=l2

Figure 17. Excerpt from Context Extension for the Object-Role Modeling approach [28]

such as encapsulation, reuse, or inheritance. The objects can represent different context types and provide interfaces for the retrieval and processing of their context information. Hofer et al. used such object oriented models for the Hydrogen context framework [6].

In 2002, Henricksen et al. [29] presented an object-oriented approach which is the predecessor to their extension for the ORM approach. They already have a graphical representation, as depicted in Figure 18. Henricksen et al. use classifications for context modeling, such as "static" (i.e., "remain fixed over the lifetime of the entity") or "dynamic" associations. They split the dynamic associations into "sensed", "derived", and "profiled".

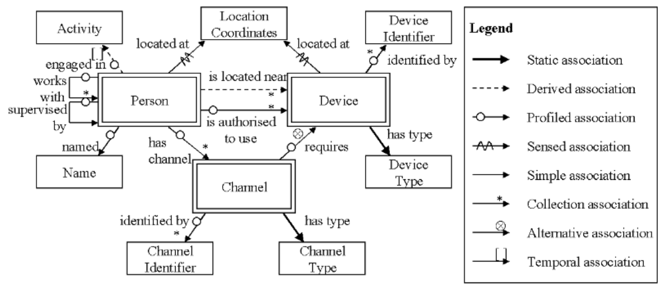


Figure 18. Example for an Object Oriented Context Model taken from [29]

Logic Based Models use formal methods to specify context information and rules that can be applied to them. Hence, they usually provide a high degree of formality. Typically, in the reasoning process new facts can be derived based on known facts and a given set of deduction rules. Albeit being very formal and precise, profound logic-based modeling is quite hard and modeling given facts can become very complex. One such approach has been published in 1994 by McCarthy and Buvac [30].

Ontology Based Models are used to represent concepts and interrelations. They are a very promising instrument for context modeling, especially with the option to apply ontology reasoning techniques and automatic derivation of new relationships. A representative of this class of models is the Aspect-Scale-Context (ASC) model (shown in Figure 19), which is based on the Context Ontology Language proposed by Strang et al. [31]. According to this model, an *Aspect* has one or more *Scales*, and a *Scale* has one or more *ContextInformation* items. These model elements are "interrelated via *hasAspect*, *hasScale* and *constructuredBy* relations" [31].

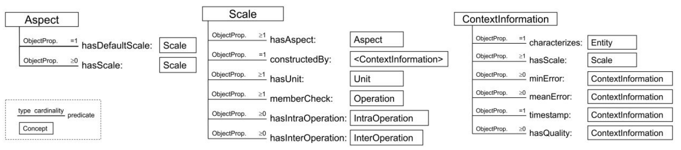

Aspect
ObjectProp. =1 hasDefaultScale: Scale
ObjectProp. ≥0 hasScale: Scale
type cardinality predicate
Concept
Scale
ObjectProp. ≥1 hasAspect: Aspect
ObjectProp. =1 constructedBy: <ContextInformation>
ObjectProp. ≥1 hasUnit: Unit
ObjectProp. ≥1 memberCheck: Operation
ObjectProp. ≥0 hasIntraOperation: IntraOperation
ObjectProp. ≥0 hasInterOperation: InterOperation
ContextInformation
ObjectProp. =1 characterizes: Entity
ObjectProp. ≥1 hasScale: Scale
ObjectProp. ≥0 minError: ContextInformation
ObjectProp. ≥0 meanError: ContextInformation
ObjectProp. =1 timestamp: ContextInformation
ObjectProp. ≥0 hasQuality: ContextInformation

Figure 19. Example for an Ontology Based Context Model taken from [31]

For further details, the reader is referred to two surveys: Baldauf et al. [23] survey existing context systems and frameworks, including their respective context models. Another survey by Bettini et al. [32] describes the state of the art in context modeling and reasoning. In summary, the decision which kind of modeling approach to choose can only be made by investigating the underlying application scenario and the context to be modeled. Regardless of the presented approaches, none of them consider the uncertainty in context detection and are thus unsuited for security purposes.

In our work, we use a combination of the presented context modeling approaches (hybrid approaches) and extend them with two quality attributes to deal with uncertainty. The introduced quality attributes improves the context detection to be reliable and secure. Hence, our approach is also suited for security purposes and can improve security decisions.

C. Context-awareness Enhancing Mobile Device Security

There exist several frameworks for enhancing systems and especially Android with context awareness. We will provide an overview by presenting solutions especially for enhancing mobile device security.

Context-Related Policy Enforcement (CRePE), developed by Conti et al. [33], is a context-aware framework that enhances mobile device security. The approach is to hook into Android's permission checking mechanism and to take dynamic security decisions based on context information. Moreover, CRePE may perform different actions (e.g., system shutdown) specified in the user-definable policies. To enable dynamic permission checks, CRePE makes modifications to the Android system. The context check is done in the *PolicyManager* component that interacts with the *CRePE PermissionChecker* and the *ActionPerformer* component (see Figure 20).

CRePE offers no context model. However, Conti et al. define policies that include rules and context information (only location). In addition, priorities for rules and the handling of conflicts are described in [33].

CRePE improves security of the Android system by making permission checks context-aware, and it adds additional context-aware features, such as actions controlled by policies.

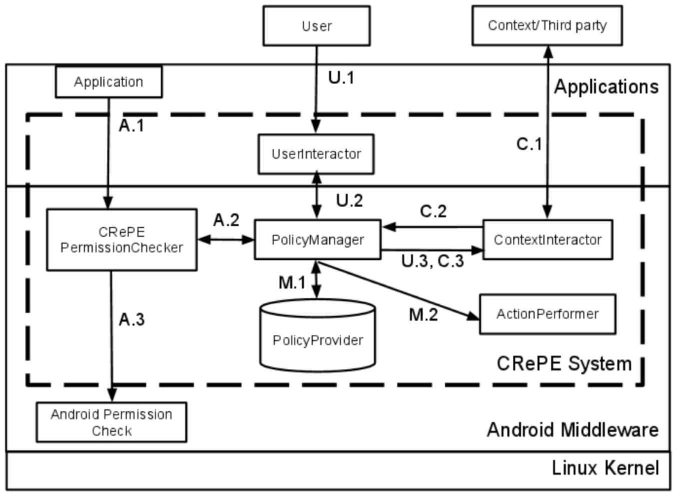


Figure 20. CRePE Architecture [33]

Context-Aware Usage Control for Android (ConUCON) by Bai et al. [34] is an extension of the UCON model [35] for Android. The framework consists of different components such as Policy Enforcement Point (PEP), Policy Decision Point (PDP), Policy Information Point (PIP) and a Policy Administration Point (PAP), as depicted in Figure 21. The components and their behavior relates to XACML [36], which is a standard describing declarative access control policies and a processing model for it.

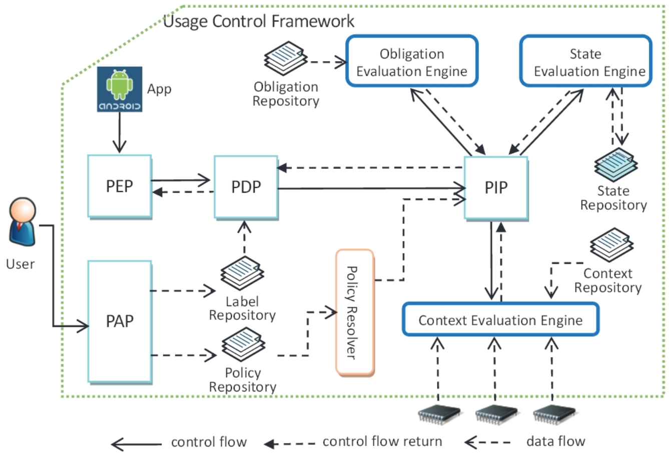


Figure 21. ConUCON Framework [34]

Contexts are part of these policies and are specified by using the tags *Context*, *ContextComposition*, and *Factor*. A context constraint is broken down into different *ContextCompositions*, which are always connected with a Boolean $\wedge$. The composition defines a logical operator (i.e., $\wedge$, $\vee$, $\neg$), which is used to aggregate the subordinated factor-tags. Finally, a factor specifies the context information such as time, battery, or Wifi state. A factor can have complex expressions such as "batteryPower $\geq$ 30%" or periodic expressions (cf. [34]). The model used can be described as Boolean logic with expressions. ConUCON also improves security of the Android system by using context-aware usage control policies.

Similar to our work, CRePE [33] and the ConUCON system [34] provide a technical solution to use context information to enhance security of mobile devices. However, CRePE uses only location as context information. Contrary to our approach, they do not address reliability and security of the context detection. In addition, they cannot deal with uncertainty of context information.

VII. Conclusion and Future Work

We presented a model for representing security-relevant contexts as context descriptions. The model contains a security rating for each evaluator to quantify the overall trustworthiness of the context description during runtime. In addition, the model provides a relevance rating expressing the conduciveness of a context information source to the overall context. The context descriptions enable context-aware security decisions by referencing them as a decision criterion in security policies.

We applied the context detection mechanism in an industrial scenario and showed its general applicability. The Mobile Device Management (MDM) solution can be enriched with contextual information to improve its decision making. We have taken some lessons learned from this industrial application, which we used to improve the generation of our context description.

In a second evaluation, we used the context-aware IND^{2}UCE framework for Android to analyze the improvement in context detection. The overall correctness of our context detection ranges from 55% to nearly 73%. But, the precision optimized context descriptions for the $0 \rightarrow 1$-transitions reached a correctness of 100% for one participant (best case) and nearly 96% for the participant with worst results. These values fulfilled our expectations and they are promising for being used in security related settings. However, the generalizability has to be shown in future work.

Future work will investigate potentials to return additional data types as an overall context result. Enriched context types facilitate the use of contextual information in the decision making process and improve expressiveness for security policy specifications. Presently, context descriptions are specified manually before being activated and are therefore rather static. Future work will investigate how context descriptions can be parametrized at runtime. This may include the use of context results as parameters for other context descriptions.

Regarding the security and relevance rating, we will further extend our evaluation criteria. We realized that faking the presence of contextual information can be easier in some cases then faking its absence (e.g., it is easier to simulate the SSID of a wireless access point than jamming the beacons from an existing one). Finally, we will explore the inclusion of accuracy information into our model as an additional quality attribute for judging the reliability of context information.

Acknowledgment

This paper is a result of the Software Campus project KoSiUX (Kontextsensitivität zur Steigerung der Sicherheit und User-Experience mobiler Anwendungen), which is funded by the German Ministry of Education and Research, grant number 01IS12053, and the European project SECCRIT, funded by the European Commission in the context of the Research Framework Program Seven (FP7), grant agreement number 312758. The authors are responsible for their content.

References

[1] C. Jung, A. Eitel, D. Feth, and M. Rudolph, “Dealing with uncertainty in context-aware mobile applications,” in MOBILITY 2015: The Fifth International Conference on Mobile Services, Resources, and Users, June 2015, pp. 1–7.

[2] S. I. A. Shah, M. Ilyas, and H. T. Mouftah, Pervasive Communications Handbook, 1st ed. Boca Raton, FL, USA: CRC Press, Inc., 2011.

[3] B. Schilit, N. Adams, and R. Want, “Context-aware computing applications,” in Proceedings of the Workshop on Mobile Computing Systems and Applications. IEEE Computer Society, 1994, pp. 85–90.

[4] Óscar D. Lara and M. A. Labrador, “A survey on human activity recognition using wearable sensors,” Communications Surveys Tutorials, IEEE, vol. 15, no. 3, Third 2013, pp. 1192–1209.

[5] S. Wang and G. Zhou, “A review on radio based activity recognition,” Digital Communications and Networks, vol. 1, no. 1, 2015, pp. 20 – 29. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352864815000115

[6] T. Hofer, W. Schwinger, M. Pichler, G. Leonhartsberger, J. Altmann, and W. Retschitzegger, “Context-awareness on mobile devices - the hydrogen approach,” in Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS’03) - Track 9 - Volume 9, ser. HICSS ’03. Washington, DC, USA: IEEE Computer Society, 2003, p. 292.1.

[7] Android Developers. Location Strategies. http://developer.android.com/. [Online]. Available: https://developer.android.com/guide/topics/location/strategies.html [retrieved: May, 2016]

[8] C. Jung, D. Feth, and Y. Elrakaiby, “Automatic derivation of context descriptions,” in 2015 IEEE International Inter-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), March 2015, pp. 70–76.

[9] MIT Media Labs. funf Open Sensing Framework. http://funf.org/. [Online]. Available: http://funf.org/developers.html [retrieved: January, 2016]

[10] funf.org. Funf journal. https://play.google.com/store/apps/. [Online]. Available: https://play.google.com/store/apps/details?id=edu.mit.media.funf.journal [retrieved: January, 2016]

[11] Gridvision Engineering GmbH. Gleeo time tracker. https://play.google.com/store/apps/. [Online]. Available: https://play.google.com/store/apps/details?id=ch.gridvision.pbtm.androidtimerecorder [retrieved: January, 2016]

[12] C. Jung, D. Feth, and C. Seise, “Context-aware policy enforcement for android,” in Software Security and Reliability (SERE), 2013 IEEE 7th International Conference on, June 2013, pp. 40–49.

[13] Android Developers. Device Administration. http://developer.android.com/. [Online]. Available: http://developer.android.com/guide/topics/admin/device-admin.html [retrieved: January, 2016]

[14] P. Brown and J. Bovey, “Context-aware applications: from the laboratory to the marketplace,” IEEE Personal Communications, vol. 4, no. 5, 1997, pp. 58–64.

[15] R. Hull, P. Neaves, and J. Bedford-Roberts, “Towards situated computing,” in Proceedings of the 1st IEEE International Symposium on Wearable Computers, ser. ISWC ’97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 146–153.

[16] D. Franklin and J. Flachsbart, “All gadget and no representation makes jack a dull environment,” In Proceedings of AAAI 1998 Spring Symposium on Intelligent Environments. AAAI TR SS-98-02, 1998, pp. 1–6.

[17] A. Ward, A. Jones, and A. Hopper, “A new location technique for the active office,” Personal Communications, IEEE, vol. 4, no. 5, 1997, pp. 42–47.

[18] N. S. Ryan, J. Pascoe, and D. R. Morse, “Enhanced reality fieldwork: the context-aware archaeological assistant,” in Computer Applications in Archaeology 1997, ser. British Archaeological Reports, V. Gaffney, M. van Leusen, and S. Exxon, Eds. Oxford: Tempus Reparatum, Oct. 1998, pp. 269–274.

[19] J. Pascoe, “Adding generic contextual capabilities to wearable computers,” in Proceedings of the 2nd IEEE International Symposium on Wearable Computers, ser. ISWC ’98. Washington, DC, USA: IEEE Computer Society, 1998, pp. 92–99.

[20] J. Pascoe, N. Ryan, and D. Morse, "Issues in developing context-aware computing," Handheld and ubiquitous computing, 1999, pp. 208–221.

[21] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a Better Understanding of Context and Context-Awareness," in HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing. London, UK: Springer-Verlag, 1999, pp. 304–307.

[22] T. Strang and C. Linnhoff-Popien, "A Context Modeling Survey," in Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England, 2004, pp. 1–8.

[23] M. Baldauf, S. Dustdar, and F. Rosenberg, "A survey on context-aware systems," International Journal of Ad Hoc and Ubiquitous Computing, vol. 2, no. 4, June 2007, pp. 263–277.

[24] N. Ryan, "ConteXtML: Exchanging Contextual Information between a Mobile Client and the FieldNote Server," in Computing Laboratory, University of Kent at Canterbury, CT2 7NF, UK, August 1999, pp. 1–8.

[25] M. Samulowitz, F. Michahelles, and C. Linnhoff-Popien, "Capeus: An architecture for context-aware selection and execution of services," in New Developments in Distributed Applications and Interoperable Systems, ser. IFIP International Federation for Information Processing, K. Zieliński, K. Geihs, and A. Laurentowski, Eds. Springer US, 2002, vol. 70, pp. 23–39.

[26] Q. Z. Sheng and B. Benatallah, "Contextuml: A uml-based modeling language for model-driven development of context-aware web services development," in Proceedings of the International Conference on Mobile Business, ser. ICMB '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 206–212.

[27] J. Bauer, "Identification and modeling of contexts for different information scenarios in air traffic," 2003.

[28] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Generating context management infrastructure from high-level context models," in In 4th International Conference on Mobile Data Management (MDM) - Industrial Track, 2003, pp. 1–6.

[29] K. Henricksen, J. Indulska, and A. Rakotonirainy, "Modeling context information in pervasive computing systems," in Proceedings of the First International Conference on Pervasive Computing, ser. Pervasive '02. London, UK, UK: Springer-Verlag, 2002, pp. 167–180.

[30] J. McCarthy and S. Buvac, "Formalizing context (expanded notes)," Computer Science Stanford University, Stanford, CA, USA, Tech. Rep., 1994.

[31] T. Strang, C. Linnhoff-Popien, and K. Frank, "Cool: A context ontology language to enable contextual interoperability," in LNCS 2893: Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003). Volume 2893 of Lecture Notes in Computer Science (LNCS)., Paris/France. Springer Verlag, 2003, pp. 236–247.

[32] C. Bettini, O. Brdiczka, K. Henricksen, J. Indulska, D. Nicklas, A. Ranganathan, and D. Riboni, "A survey of context modelling and reasoning techniques," Pervasive and Mobile Computing, vol. 6, no. 2, 2010, pp. 161–180.

[33] M. Conti, V. T. N. Nguyen, and B. Crispo, "Crepe: context-related policy enforcement for android," in Proceedings of the 13th international conference on Information security, ser. ISC'10. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 331–345. [Online]. Available: http://dl.acm.org/citation.cfm?id=1949317.1949355

[34] G. Bai, L. Gu, T. Feng, Y. Guo, and X. Chen, "Context-aware usage control for android," in SecureComm, 2010, pp. 326–343.

[35] J. Park and R. Sandhu, "The UCON ABC usage control model," ACM Trans. Inf. Syst. Secur., vol. 7, no. 1, February 2004, pp. 128–174.

[36] OASIS, "extensible access control markup language (xacml) version 3.0 - committee specification 01," http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-cs-01-en.pdf, August 2010.