Robustness in Context-Aware Mobile Computing

Hannes Wölf, Klaus Herrmann and Kurt Rothermel
Institute of Parallel and Distributed Systems, Universität Stuttgart, Germany
Email: {*hannes.wolf*|*klaus.herrmann*|*kurt.rothermel*}@ipvs.uni-stuttgart.de

***Abstract*—High level context recognition and situation detection are enabling technologies for unobtrusive mobile computing systems. Significant progress has been made in processing and managing context information, leading to sophisticated frameworks, middlewares, and algorithms. Despite great improvements, context aware systems still require a significantly increased recognition accuracy for high-level context information on uncertain sensor data to enable the robust execution of context-aware applications. Recently Adaptable Pervasive Workflows (APF)s have been presented as innovative programming paradigm for mobile context-aware applications. We propose a novel Flow Context System (*FlowCon*) that builds upon APFs. FlowCon uses structural information from the APF to increase accuracy of uncertain high-level context information up to 49%. This way we make an important step to enable robust execution of mobile context-aware applications.**

***Index Terms*—context-aware mobile computing, flows, uncertain high-level context information, robustness**

I. Introduction

In general, context information is one of the most important information sources for mobile computing applications and the development of mobile devices and sensors has made them widely available. But the acquisition of sensor data inherently introduces uncertainty into the system that applications have to deal with. A great number of context management systems have been developed that provide context information to applications considering different application areas such as sensor networks, mobile computing, smart homes, and even systems suited for global context management [1]–[3]. These systems are built to handle a great variety of different context information, ranging from primary context such as location, time and identity of objects, to complex context information based on ontologies and context reasoning [4].

Usually, context management systems have to measure context information from the real world using sensors. Sensor data is always uncertain, but can be quantified by accuracy and precision. While accuracy defines how close the current reading represents the actual measured value, precision denotes the statistical deviation when the measurement is repeated under unchanged conditions. The more accurate and precise the measurement is the less uncertain the measured information is. Acquired context information is then processed, aiming for two goals. 1) Reduce uncertainty using multiple readings or sensor fusion techniques. 2) Combine context information to get more abstract high-level context information based on context reasoning, situation detection or activity recognition. High-level context information is less certain, because it is extracted from already uncertain data. The context management system then delivers the context information to the single applications, which should handle the uncertain information.

However, context management systems do not deal with uncertainty of context information at all [1], or provide only uncertainty handling for rather simple low-level context information [5], [6]. But certain high-level context information is necessary for the correct execution of context-aware mobile applications. Actually, none of the existing systems allows the applications to assist the context management system to deal with uncertainty of abstract high-level context information.

We propose an algorithm to increase the accuracy of high-level context information using Adaptable Pervasive Flows (APF) or shortly *flows*. Flows originate from classical workflows [8], and were recently proposed as a programming paradigm for pervasive applications [7] suitable for execution on mobile devices. A flow basically consists of a set of activities that are glued together by transitions which define the execution order of the activities. An activity in a flow either represents some computational task, like writing a database record or calling a Web Service, or it describes a task that a human has to perform in the real world. In the latter case the successful flow execution depends on a context management system that recognizes the activities of the human, when the tasks are actually performed.

As main contribution of this paper, we propose the *Flow Context System* (*FlowCon*). FlowCon is a context manager that performs workflow mining on the flow structure to derive dependencies between context information which drive the flow execution. The gained knowledge is used to increase the recognition accuracy of the related human activities, thus decreasing their uncertainty. This way the flow execution becomes more robust i.e. even when the original accuracy of the available context information is low the flow executes correctly.

We evaluate the FlowCon algorithm using real traces gathered in a health-care environment, and show through simulation that we can increase the accuracy of the context information delivered to the application up to 49%. The overall robustness of the system can also be increased significantly, despite the very low accuracy of the activity recognition results from the traces.

The rest of the paper is structured as follows: In Section II, we discuss previous approaches that are related to workflows and context awareness and then introduce our application scenario in Section III. Subsequently, we describe FlowCon, including our formal flow model specification and the FlowCon algorithm in Section IV. We then discuss the evaluation results in Section V. Finally, we conclude the paper with a summary and an outlook in Section VI.

II. Related Work

As our approach is based on flows, we investigate existing work on context-aware and mobile workflow management. Furthermore, we discuss the relation of FlowCon to the area of workflow mining.

Mobile workflow execution is a key technology to enable the use of flows on mobile devices. A feasible approach to execute workflows on mobile devices has been presented by Hackmann et al. [9], [10]. But the mobile workflows considered do not take context information into account and thus they are suited for classical workflows only.

For the classical workflows, context integration was first introduced by Wieland et al. [11], where the authors provide certain operators to access context provided by a context management system. The approach was later extended to deal with Quality of Context [12]. A policy language is used to define the amount of uncertainty in context information the workflow accepts and filter out context information that does not match this filter. This allows the flow to omit context information that may lead to wrong execution. But there is no algorithm or evaluation provided. Fuzzy Workflows, have been presented by Adam et al. [13], [14] A Fuzzy Workflow is able to make decisions based on fuzzy input information. The input from different sources is fed into a fuzzy logic operator within the flow and the result leads to a clear decision which is used to continue the workflow execution accordingly. However, these approaches are extensions for classical workflows, which are usually not flexible enough for execution in a mobile context-aware environments. But even more importantly, they do not enable the workflow to contribute in decreasing uncertainty of context information.

The PerFlows presented by Urbanski et al. [15] are context-aware, suitable for pervasive scenarios and provide flexible activity scheduling and processing. However, they require heavy user interaction to work properly. In our previous work [16], we presented an approach for dynamic context-awareness suited for pervasive flow-based applications. Both approaches neglect the handling of uncertain context information.

Workflow Mining comes in two different flavors. On the one hand, a new workflow is created from event logs of different applications in order to visualize the actual flow of work and possibly automate the created workflow using classical workflow management techniques. On the other hand existing workflows can be used to extract knowledge. While we also extract knowledge from flows, there are no approaches that improve context processing this way. Buffett and Geng [17] have proposed to label the activities of workflows by learning from event logs. This approach is somewhat similar to ours. It uses learning with Bayesian Networks, resolves the ordering of activities in the generated workflows and analyzes the paths taken in workflow execution. However the algorithm is applied to collected event logs and the workflows are mined in with an offline algorithm. Furthermore, it assumes that the log data contains no uncertain information, e.g., no real-world context is taken into account. In contrast, we know the flow structure and learn the event dependencies at runtime, taking uncertain context information into account.

In summary, classical context-aware workflows and workflows tailored for pervasive computing provide little mechanisms to deal with uncertain context information. Furthermore no system discussed here uses the knowledge encoded in the workflow to improve the accuracy of context recognition, thus reducing the amount of uncertainty the flow actually has to deal with. To the best of our knowledge FlowCon is the first system to achieve this.

III. Scenario Description

To evaluate FlowCon, we used a set of data collected in a geriatric nursing ward. This is an intensive care station for elderly people suffering from dementia and similar old-age diseases. Each of the patients there needs help around-the-clock. There are well-defined medical guidelines for accomplishing the daily work within the ward. This scenario provides a relatively limited and basically fixed set of nurses and patients and the process structure is also quite stable. All activities performed (e.g. treatment, medication) stringently have to follow the guidelines and the results of some must be documented. But, there is rich human interaction between nurses and patients. In practice the nurses do not have the time to document all their activities properly, even when using a mobile device. This necessitates the use of activity recognition systems to sense which tasks the nurses actually accomplish. These properties assure that the scenario is suitable for testing our system.

The traces we obtained from the nursing ward follow the daily morning routine of a single nurse. Each nurse was given a mobile phone which she wears in her coat pocket for data collection. The sensor readings available in the traces are (1) received WiFi signal strength, (2) measured magnetic field strength, (3) measured acceleration and (4) recorded sound The WiFi readings were used to estimate the indoor position of the nurse on a room-level granularity, the magnetic field sensor for facing direction. The necessary WiFi infrastructure was already present in the hospital, so there was no need to deploy further infrastructure. The acceleration data were used to do activity recognition like mode of locomotion. The recorded sound snippets were also used to classify activities according to typical background noises like the sound of a shower when a nurse is helping a patient taking a shower. For more complex context information, multiple of the mentioned modalities were used for recognition. The collected data has been manually labeled for a training set, but there is also an unlabeled test set.

Each trace covers about 2 and a half hours, where the nurse had to care for a total of three patients. The basic support for every patient is very similar and consists of four distinct steps. The (1) morning examination includes measuring the pulse and the blood pressure of the patient. Blood samples are taken regularly once or twice a week per patient. During the

(2) morning hygiene, the nurse helps the patient with getting up, washing and dressing. Following that the nurses help the patients having their (3) breakfast. Finally she supervises and assists the patient taking his (4) daily morning medication according to the patients capabilities.

This process consists of a number of more concrete tasks. While most of these tasks are accomplished every day, each nurse flexibly alters the execution order. In order to limit these flexible changes, we focus the process execution on the blood sample examination process. In the following, we describe the process guideline in detail and point at the possible execution variations that may happen. Those variations are important for our algorithm design because the knowledge we extract is influenced by the habits of the nurse.

When a blood sample examination is scheduled for a patient—this is documented in the patient record—the nurse takes it after the daily measurement of pulse and blood pressure. A reusable butterfly needle is used, because there are taken up to four blood samples in a row. A formal representation of the flow is depicted in Figure 1 but some of the details there will be explained later. To obtain a blood sample the nurse has to perform the following activities. First, she (a_1) fastens a cuff to the upper arm of the patient. She then starts (a_2) searching a vein for setting the butterfly. After that, she (a_3) unpacks the butterfly and (a_4) disinfects the elbow pit. She punctures the patient (a_5) setting the butterfly and (a_6-a_9) takes the samples. Finally, she (a_{10}) labels each sample with the patients credentials.

While getting the blood sample from the patient, the nurse basically has two variation options. She can either disinfect the elbow pit first and then unpack the butterfly, or the other way around. However she must complete both activities before she can set the butterfly. Moreover, she is free to chose the order in which the individual samples are taken when she has set the butterfly.

The mentioned variations lead to interesting questions. When the activity a_2—search vein—has been recognized, the context system cannot know which will be the next activity that the nurse executes. Because of this, it is much harder to recognize the following activity compared to a scenario with a predefined fixed sequence of activities. In this case FlowCon is able to increase context recognition performance. Furthermore, when the flow execution waits for an activity a_5—set butterfly—which depends on more than one previous activity, the recognition of the preceding activities (a_3, a_4) increases the probability that the next recognized activity will likely be a_5.

The correct flow execution leverages automatic documentation of the blood sample taking and relives the nurse of some of the paperwork. But this is a tough task, because it requires to recognize the activities for every single step just using the uncertain activity recognition results to drive the workflow execution.

IV. FlowCon

In this section, we introduce FlowCon. At first, we define the formal context model as foundation for the algorithm. Next, we present our formal *flow model*, which allows the application programmer to create flows and to specify the context information that is relevant to the flow. We then explain the execution semantics for flows and finally describe the FlowCon algorithm itself.

A. Context Model

As we motivated in the introduction, the flow execution is driven by the recognized activities performed by human users. In order to execute pervasive applications based on flows, we need to define the representation of recognized activities and uncertainty. The activities we consider in terms of our hospital scenario have been described in Section III. For successful recognition, context information from a multitude of sensor readings and also from different modalities that are all uncertain have to be composed. For practical usage, neither the accuracy nor the precision of situation detection is adequate. The sampling rate can hardly be increased due to data overload and higher processing costs. Furthermore, the accuracy of the results decreases significantly, depending on the situation and the used detection and classification methods. Another drawback of situation detection is the high effort spent for training the system. Because of this, feasible situation detection is only possible for specific and well defined application areas. However our hospital scenario does not fulfill this requirement and situation detection is only applicable with low overall accuracy.

In the following we assume that the number of situations to detect in our hospital scenario is finite in practice. Based on this, we define our representation of *events*.

***Definition** 4.1 (Event):* A situation that can be acquired from the environment is referred to as event $e \in U_e$, where U_e denotes the universe of all events that the context system can measure.

In our blood sample flow we have events for the following situations: (e_1) Apply Cuff, (e_2) Search Vein, (e_3) Disinfect Elbow Pit, (e_4) Unpack Butterfly, (e_5) Set Butterfly, (e_6) Get Blood Sample and (e_7) Label Samples. Each event indicates that the corresponding task has been performed in the real world. When compared to the blood sample flow (cf. Figure 1) it can be seen that there is a one-to-one mapping between events and activities for the scenario. It is of course possible to have more than one event indicating the completion of an activity in the flow. We discuss the details when we introduce the flow execution semantics later in this section. Each event e is a member of at least one *event type*.

***Definition** 4.2 (Event Type):* An event type $E \subset U_e$ contains a number of individual events $E := \{e_1, \ldots, e_n\}$.

The event type is a tool for the application designer. When the application is created, the individual events e are grouped together to certain types E which correspond semantically. For the blood sample flow, we define only a single event type $E_b := \{e_1, \ldots, e_7\} \subset U_e$ that contains all events that should happen, during its execution. The purpose of an event type is twofold: Firstly it allows the application designer to react to multiple related events in a distinct fashion. For example, the events of E_b indicate that something has happened that is important for the blood sample flow. Secondly the related semantics of the grouped events allow a more accurate recognition and classification. Other events that are not contained in the expected event type are likely out of scope. When the flow executes a certain activity, it registers the event type and receives a notification from the context recognition system. The notification contains an *event instance* that indicates which situation has been detected by the context system.

***Definition** 4.3 (Event Instance):* Let $E := \{e_1, \ldots, e_m\}$ be an event type. An event instance $I_E : E \rightarrow [0, 1]$ defines a probability distribution function over all events for the specified event type, i.e. $\sum_{e \in E} I_E(e) = 1$. Further let $\mathcal{P}(I_E)$ denote the set of all possible event instances for a given event type E i.e. all possible distributions.

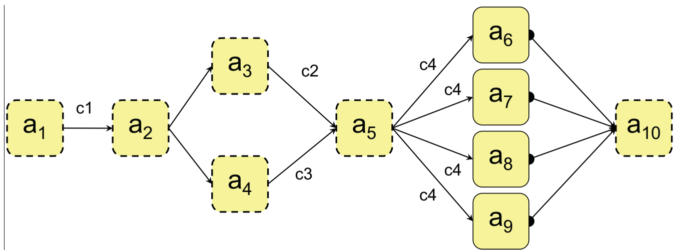
The definition of the event instance probability distribution indicates that we decided to use Probability Theory to represent uncertainty for our events. Other uncertainty models like Possibility Theory [18] or Dempster-Shafer Theory of Evidence [19] have also been considered. There were two reasons to use Probability Theory instead. Firstly, the interpretation of those other models is not trivial and can cause unwanted side effects, such as counter-intuitive results when combining uncertain events. Probability Theory is simpler, but it leaves little space for interpretation, thus easing decision making. Secondly, the other uncertainty models provide additional information, but also have stricter requirements regarding the context system. In order to provide meaningful probability distribution functions for the event instances, the context system has to provide statistical data on sensor information such as trustworthiness of a given modality or a certain sensor. Those values were not available from the given traces and the number of traces was to small to extract useful values on our own (c.f. Section V).

B. Flow Modeling

A *flow model* is a template for a single running flow based application. The application programmer defines all the *activities* and their partial ordering using *transitions*. *Conditions*, that are annotated to the transitions, further influence the ordering. When the flow is actually executed an instance is created and supplied with additional information. For example, information of the actual nurse and patient are provided to the specific blood sample flow (cf. [16]). We explain each component of the flow model and subsequently define the execution semantics.

***Definition** 4.4 (Flow Model):* A flow model $\mathcal{F}$ is a 4-tuple $\mathcal{F} := (A, T, C, \mu)$, consisting of a set of activities A, a set of transitions T, a set of conditions C and a transition marker function μ.

The flow model $\mathcal{F}$ specifies a directed acyclic graph with activities as vertexes and transitions as edges. The set of conditions as well as the transition marker function add information to the single transitions.



a_1 apply Cuff	a_3 unpack butterfly	a_5 set butterfly	a_7 sample 2	a_9 sample 4
a_2 search vein	a_4 disinfect elbow pit	a_6 sample 1	a_8 sample 3	a_{10} labeling

Fig. 1. Blood Sample Flow

***Definition** 4.5 (Activity):* An activity a represents an atomic piece of work within a flow. This includes invoking external (possibly human) services as well as internal computations. The set $A := \{a_1, \ldots, a_n\}$ defines all activities of a flow. An arbitrary number of event types can be added to each activity. Let $\epsilon_a : \mathbb{N} \rightarrow \mathcal{P}(U_e)$ be the event type assignment function for a, where $\mathcal{P}(U_e)$ denotes powerset over the universe of events. Further, let k be the number of event types associated to a then, $\epsilon_a(i)$ yields the corresponding event type for $i \leq k$, and $\emptyset$ otherwise. Furthermore activities may be marked as mandatory.

The activities $A_b := \{a_1, \ldots, a_{10}\}$ of the blood sample flow are depicted in Figure 1. As the nurse executes every activity, each one depends on context information, hence the event type E_b is associated with every activity, i.e. $k = 1$ and $\epsilon_a(1) = E_b$ for $a \in A_b$. Furthermore all the activities but $a_6 - a_9$ are mandatory in order to complete the flow execution successfully. The mandatory activities depicted in Figure 1 have a dashed border.

***Definition** 4.6 (Transition):* Given a set of activities A, the set of all transitions within a flow is $T \subseteq A \times A$. A transition $t = (a_x, a_y)$ represents a directed control flow dependency from a_x to a_y with $a_x, a_y \in A$. A transition is annotated with exactly one transition condition, that is referred to as t.c. Further, we define $d_{in}(a_i) := |\{(a_x, a_y) \in T | a_i = a_y\}|$ and $d_{out}(a_i) := |\{(a_x, a_y) \in T | a_i = a_x\}|$ as degree of incoming and outgoing transitions for an activity.

The transitions allow certain control flow variants (cf. Figure 1): linear sequences ($d_{out}(a_1) = 1$), parallel branching ($d_{out}(a_2) > 1$) and joins like for ($d_{in}(a_4) > 1$), and combinations of those. Conditional decisions can be made taking the transition conditions into account.

***Definition** 4.7 (Condition):* A condition c is inductively defined as $c \rightarrow (I_E(e)) | (c_1 \vee c_2) | (c_1 \wedge c_2) | \neg(c_1)$ with $e \in E$, $c_1, c_2 \in C$ and the common semantics for the probabilistic logical operators. C is then the set of all conditions within a flow model.

Please note that evaluation semantics for the events $I_E(e)$ and the logical operators depend on the used uncertainty model. Using Probability Theory, we evaluate the conditions with plain probabilistic logic (cf. [20]). Conditions allow control flows like conditional branching a_5 and forks a_2 (cf. Figure 1). We used the following four transition conditions in the blood sample flow. The condition c_1 validates that the cuff has actually been applied correctly before the nurse starts searching a vein, which may be impossible otherwise. c_2 and c_3 guarantee that the disinfect elbow and unpack butterfly have been completed, maintaining hygiene guidelines, before it is allowed to puncture the patient. Finally, c_4 checks that no blood sample is taken before the butterfly actually has been set up correctly. The corresponding transitions in Figure 1 are annotated with the condition identifier.

In order to specify well-defined execution semantics for our flow model, we further introduce transition markers.

***Definition** 4.8 (Transition marker):* The transition marker function $\mu := T \rightarrow [true, false]$ assigns markers to all transitions in the flow, where $\mu(t) = true$. If a transition has a marker, the execution of this transition is not required to be active in order to start the target activity of the transition.

The transition markers allow joins of multiple flow branches where not all branches must or can be executed during a single flow execution. As we explained in the scenario description, the number of taken blood samples varies. Therefore all the incoming edges of the final activity a_{10} have a marker (c.f. Figure 1). This way the execution of the activity is possible, when at least one of the previous activities (a_6-a_9) has been completed. The markers are denoted as dots at the origin of transitions.

We now present the execution semantics of a flow model. On instantiation the flow model, starts with the execution of the activities, where $d_{in}(a) = 0$. When an activity is executed it subscribes for its event types $\epsilon_a(i), i \in \{1, \dots, k\}$ at the context system. After an event instance I_E has arrived for every event type, the activity is completed and the outgoing transitions are checked. The transition conditions $t.c$ are evaluated as defined earlier. When the resulting probability is above a certain *navigation threshold* t_n, the condition becomes true and the transition is also activated. The navigation threshold defines the minimal probability a certain condition must have. Assume the execution of a_5 has just been completed and the received event instance $I_{E_b}(e_5) = 0.47$ indicates that the butterfly was set. Given a navigation threshold $t_n = 0.4$ would allow the transition to trigger the target activity. When the navigation threshold would be higher, e.g. $t_n = 0.5$, then the flow execution would not be continued because the situation detection result was too uncertain. A subsequent activity begins execution if all incoming transitions are activated ($t.c \geq t_n$) or marked ($\mu(t) = true$). If all transitions are marked at least one has to be activated.

The execution of the flow model yields a flow trace. When an activity is completed, this is recorded in the flow trace along with the event instances it received.

***Definition** 4.9 (Flow Trace):* A flow trace $\mathcal{T}$ is a sequence of completed activities $\mathcal{T} := (a_1, \dots, a_k)$ ordered increasingly by completion time with $k \leq |A|$. The event instances each activity has received are also stored within the trace. The function $\theta_a : \mathcal{P}(U_e) \rightarrow \mathcal{P}(I_E)$ yields the received instance for an event type, if the event type is associated to the activity a.

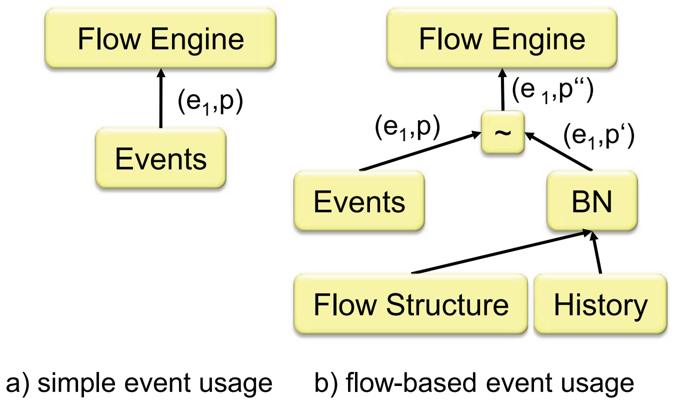


Fig. 2. Architecture Comparison

From a single trace, it is possible to reconstruct the actual execution of a flow instance and which context information, i.e. event instances, lead to this execution. All traces are stored in a flow history documenting the executions for later analysis. We use the flow history of a flow model as the data set for training the FlowCon algorithm later.

C. Algorithm

The general goal of FlowCon is to increase the accuracy of the events the flow execution relies on. More specifically, we adjust the probability distribution of an event instance, so that the statistically most probable event will be favored. The probability of this event is increased, while simultaneously the probabilities of the possible other events are decreased. This way the accuracy of the events expected by the application will be increased and their uncertainty is decreased. We train a Bayesian Network (BN) [20] to extract the knowledge from the flow, which event currently is the most probable. BN training consists of two phases: structure learning and parameter learning. While parameter learning can be achieved efficiently from a set of given observations, learning the optimal structure of a BN from such data is NP-hard. The FlowCon algorithm avoids the structure learning problem completely. It basically works in three steps. First, it analyzes the flow structure and generates a BN structure from that analysis. In the second step, it then uses the observed data from the flow history to do the parameter learning. While both of these steps can happen offline before the actual instantiation of a certain flow, the third step—the information combination—is executed at runtime. A comparison to the naive approach is depicted in Figure 2. Usually the application has to deal with the provided probability of an event (e_1, p), but FlowCon queries the BN for the current event, using the actual execution state of the flow and the already received event instances. It then combines the event with its derived statistical probability from the BN (e_1, p') and generates a new probability for the event (e_1, p''). The probability p'' will be higher than p if e_1 is statistically more probable for the application in the given context.

1) Structure Learning: To build the structure of the BN from the flow structure we assume that there exists a dependency between the events associated to two activities a_x and a_y if there exists a transition $t = (a_x, a_y)$. For example, in terms of our blood sample flow, the occurrence of the event e_1—apply cuff—necessitates the occurrence of the event e_2—search vein—afterwards. While this dependency is simple, there are more difficult cases. When we consider forks $d_{out}(a_2) = 2$, it is not clear if there is a dependency between the events of a_2 and the events of a_3, a_4. A single nurse could disinfect the arm first every time and then unpack the butterfly. This does of course change the statistical dependency between the events. The one between e_2—search vein—and e_3—unpack butterfly— will be low, while the other between e_2 and e_4—disinfect elbow pit— will be high. When we further consider the activities with $d_{in}(a) > 1$ we see that the occurrence of an event may depended on more than one previous event. Some of the preceding events may have a strong statistical dependency, while others have no effect at all. However, we create each of the possible dependencies in the beginning and adapt their strength later in the parameter learning phase, e.g. to deal with changing behavior of nurses.

We use a Bayesian Network $BN := (N, D)$ to represent the statistical dependencies from the flows, where $n \in N$ is a node and $d \in D$ is a conditional dependency. The nodes of the BN represent discrete random variables. The state space of a node n is equal to the event type E adding a `null` class. For every event type of an activity we create a node $n = a.E$ identified by the name of the activity and the event type. After that, we add the dependencies between those nodes where the activities also have a directed dependency, or more formally:

$$((a_x, a_y) \in T) \wedge (\exists i : \epsilon_{a_x}(i) = E) \wedge (\exists j : \epsilon_{a_y}(j)) = E \Rightarrow \\ ((a_x.E), (a_y.E) \in D)$$

The result is a structured BN. Please note that the learning has been very simple, because we have the flow structure which provides us with a realistic assumption which events are related to each other. Furthermore, the structure is fixed for a specific flow model. Therefore, this step can be performed offline right after the modeling phase. However, there may be multiple instances of the BN in use for different locations where the flow is actually deployed and executed, or for different actors that are associated with the flow. For example, there could be a single network trained for every nurse to take personal habits into account when executing the flow and processing the context information.

2) Parameter Learning: The next step is to train the BN with the actual statistical dependencies that have occurred. At first the conditional probability tables (CPT)s of each node are initialized with a uniform distribution. In order to train a BN we need a training data set with observations for the value of each node. We use the flow traces that are collected for each execution of a flow as data set for training. As the event instances are stored together with the flow trace, it can be converted to an observation of the values.

As an example, we look at a trace $\mathcal{T}$ from our blood sample flow. For activity a_1 there is an event instance $I_{E_b} = \theta_{a_1}(\epsilon_a(1))$ stored in the trace. The probability distribution of this event instance I_{E_b} indicates that e_1 is the most significant event, i.e. the one with the highest probability. So for training from this trace we would set the observed value for the corresponding node $a_1.E_b$ to e_1.

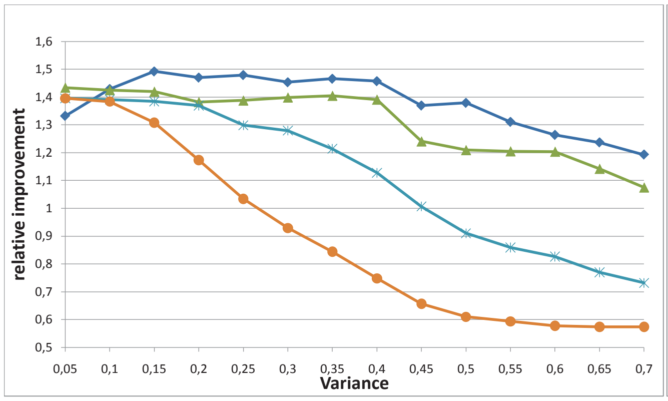
The event types associated to activities that have not been executed in a trace cannot provide event instances, thus the corresponding nodes are set to the to the `null` state for this trace. In the mentioned trace the activity a_7 may not have been executed because the corresponding blood test was not scheduled. So the value for the corresponding node $a_7.E_b$ would be `null`. The BN can be trained incrementally with the traces, as they become available. This way the CPTs are adjusted until the dependencies are appropriately represented. As we demonstrate in the evaluations a rather small training set of 25 to 50 traces is sufficient for training.

3) Information Combination: The third and final step is to retrieve the information from the BN and combine it with current context information to increase accuracy. Querying a BN usually means to initialize some of the variables with observed values and compute the conditional probabilities of the unknown variables. The observed values for the instance of the flow that is currently executed, are the past events that have already been recognized. We take all available events as evidence into account.

When the next event instance I_E arrives that only marginally denotes that the expected event has happened, we compute the conditional probability for this event using the previous event instances as current observations. The result from the BN is also a probability distribution function for the event type E. We combine this result with the probability distribution function I_E of the actual event instance. We compute the average probability for each event in the distribution and normalize the results afterwards to make the probability distribution valid again. This averaging adjusts the probability of each event by the amount of its statistical probability under the given context of the previously occurred events. Here we actually adjust the context measurement. The probability of an event that statistically occurs more often is increased, while the other way around, the probability gets lowered. Note that higher probability also means a higher accuracy for the measured event. The resulting probability distribution is stored in the event instance. The event instance is then sent to the flow instead of the original one, and processed according to the flow execution semantics.

V. Evaluation

To evaluate our system, we used the blood sample examination flow we presented in this paper. The scenario data and the flow were directly extracted from the collected real-world traces. Unfortunately, some of the event data had to be simulated and we discuss this in the setup section, along with some technical details. Following that, we present our evaluation results and analyze them.



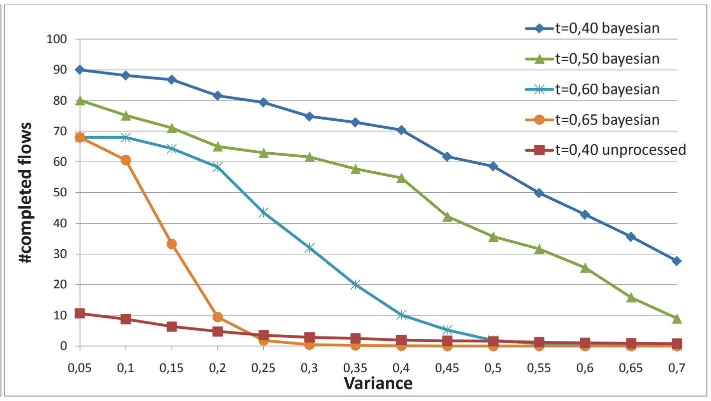


Fig. 3. Simulation Results

A. Setup

Our simulation setup consists of three main components. First a basic flow engine that is able to execute our flow models using the JUNG Framework [21]. The flow engine defines the navigation threshold t_n as described in Section IV-B. The navigation threshold is our first simulation parameter. The flow engine also implements the evaluation semantics defined for the conditions based on Probability Theory. The second component is the Bayesian Network Event Processor that implements our algorithm and processes the event instances. To represent the Bayesian Networks, we used the well-known Weka framework [22].

The third component is the context system which is responsible for feeding the context events into the flow engine. Based on the recognition results from the real traces, we generated artificial traces with the same structure and probability distributions in order to produce statistically relevant results. We first created lists of events that would allow the engine to complete the flow successfully. We did not consider out of order events or completely missing ones, but just the uncertainty of false recognitions. The event lists were then assigned with the results of the real recognition to keep the simulation as realistic as possible. The average recognition probabilities were between 40% and 60% for the correct events. But there have also been false recognitions with probabilities also up to 40%. We further added noise to those single probabilities, which effectively introduces a variance v on the absolute recognition probabilities. The resulting values were normalized to get a sound probability distribution. The variance value v is our second simulation parameter.

An experiment we simulated, consisted of the subsequent execution of 100 blood sample flows. Each experiment was repeated 25 times with the same parameter settings to achieve statistical relevance. We started with a freshly initialized BN every time and had no training data available from flow histories i.e. the parameter learning phase of our algorithm is performed online. The number of available traces for training then increases with every completed flow instance. We chose navigation thresholds between $t_n = 0.4$ and $t_n = 0.65$ with steps of 0.05 and accuracy variance values between $v = 0.05$ and $v = 0.7$ also in steps of 0.05. Please note that a variance value of $v = 0.4$ basically introduces the same amount of noise into the simulation as we have recognition probabilities from the traces. When we further increase the variance up to $v = 0.7$ this can be interpreted as feeding significantly more noise into the flow compared to the given recognition probabilities.

B. Results

We measured two properties of our system, the accuracy improvement of the events that should be delivered to the flow engine in order to allow a correct execution of the flow and the overall number of completed flows which can be interpreted as the robustness of the flow execution.

1) Event Improvement: For the event improvement we measured the relative event improvement, which is depicted in Figure 3 on the left. By relative event improvement we mean the probability of the significant event of the event instance divided by its original probability before the processing with our algorithm (p''/p c.f. Figure 2). We left out some curves for better visibility. For the thresholds $t_n = 0.4$ and $t_n = 0.5$ we observe a very good accuracy improvement performance between 49% and 39%, for variance values up to $v = 0.4$. This conditions indicate a system that has equal to higher requirements for the recognition accuracies that could actually be provided. Furthermore we can deal with a significant amount of noise quite well. However when we further increase the variance up to $v = 0.7$ the average event improvement slowly degrades to only 7%. But we still manage to improve the recognition probabilities a little. When we further increase the navigation threshold $t_n = 0.6$ and $t_n = 0.65$ the performance degrades much faster. While FlowCon is still able to achieve a good improvement for small variance values up to $v = 0.15$, we quickly get counter productive results when we further increase the variance. The break even point, where we actually make things worse using FlowCon, is $v = 0.45$ for a threshold $t_n = 0.6$ and $v = 0.25$ for a threshold $t_n = 0.65$. This strong degradation can be explained as we train the BN online during the experiments. The correct training gets more difficult and finally impossible with higher variance, values because we have fewer correct traces and the navigation threshold to achieve a correct trace is very high.

2) Flow Execution Robustness: The second observed property is ratio of flows that were executed successfully during an experiment. Those results are depicted in Figure 3 on the right. As a reference, we have also shown here the performance of our flow engine under the same conditions, but without the processing accomplished by FlowCon. Given this setting, the engine is only able to complete an average between 10.6% and 0.8% of all flows, while the combination with the FlowCon algorithm yields an average number of completed flows between 90.0% and 27.6% which is a significant improvement. The same degradation behavior as in the event quality improvement can also be observed for the overall system robustness. We perform quite well with slow degradation and a measurable drop at a variance $v = 0.45$ for the lower thresholds ($t_n = 0.4$ and $t_n = 0.5$). For the upper thresholds, the performance degrades much faster for the reasons we explained before.

VI. Conclusions and Future Work

In this paper, we have presented the Flow Context Manager (FlowCon), a novel approach based on flow-based pervasive applications that increases the accuracy of context information. Flow Evaluations, based on real-world traces gathered in a geriatric nursing home, have shown a significant increase of event accuracy of up to 49%. Furthermore, we made the execution of flow-based mobile applications more robust. The ratio of flows that could complete their execution successfully was increased significantly to up to 90% of the overall flows. The results are promising and, thus, may leverage automatic documentation in the presented scenario that frees the nurses partially of some of their paperwork.

Our approach allows flow-based pervasive applications to contribute information for the processing of high-level uncertain context information. However, the expressiveness of the used flow model is a limiting factor at the moment. We aim to extend our approach so that it can be used on more flexible flow models which can handle more complex human behavior. Consequently, we believe we can support a broader range of applications and provide them too with better activity recognition accuracy. We also want to study the use of more complex uncertainty models with our algorithm for further accuracy improvement.

VII. Acknowledgements

The work described in this paper is partially funded by the 7th Framework EU-FET Project 213339 ALLOW

References

[1] K. Henricksen and R. Robinson, "A survey of middleware for sensor networks: state-of-the-art and future directions," in *MidSens '06: Proceedings of the international workshop on Middleware for sensor networks*. New York, NY, USA: ACM, 2006, pp. 60–65.

[2] 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, pp. 263–277, 2007.

[3] R. Lange, N. Cipriani, L. Geiger, M. Grossmann, H. Weinschrott, A. Brodt, M. Wieland, S. Rizou, and K. Rothermel, "Making the world wide space happen: New challenges for the nexus context platform," *Pervasive Computing and Communications, IEEE International Conference on*, vol. 0, pp. 1–4, 2009.

[4] A. Schmidt, M. Beigl, and H.-W. Gellersen, "There is more to context than location," *Computers & Graphics*, vol. 23, no. 6, pp. 893 – 901, 1999.

[5] P. Korpip, J. Mntyjrvi, J. Kela, H. Kernen, and E.-J. Malm, "Managing context information in mobile devices," *Pervasive Computing, IEEE*, vol. 2, no. 3, pp. 42 – 51, July-Sept. 2003.

[6] R. Lange, H. Weinschrott, L. Geiger, A. Blessing, F. Dürr, K. Rothermel, and H. Schütze, "On a generic uncertainty model for position information," in *First Internationa Workshop on Quality of Context, QuaCon*, June 2009, pp. 76–87.

[7] K. Herrmann, K. Rothermel, G. Kortuem, and N. Dulay, "Adaptable Pervasive Flows–An Emerging Technology for Pervasive Adaptation," in *Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops*. IEEE Computer Society, 2008, pp. 108–113.

[8] F. Leymann and D. Roller, *Production workflow: concepts and techniques*. Prentice Hall PTR, 2000.

[9] G. Hackmann, M. Haitjema, C. D. Gill, and G.-C. Roman, "Sliver: A BPEL Workflow Process Execution Engine for Mobile Devices," in *Proceedings of 4th International Conference on Service Oriented Computing (ICSOC*, 2006, pp. 503–508.

[10] G. Hackmann, C. Gill, and G.-C. Roman, "Extending bpel for interoperable pervasive computing," in *IEEE International Conference on Pervasive Services*, Istanbul, 15-20 July 2007, pp. 204 – 213.

[11] M. Wieland, O. Kopp, D. Nicklas, and F. Leymann, "Towards context-aware workflows," in *CAiSE07 Proceedings of the Workshops and Doctoral Consortium*, B. Pernici and J. A. Gulla, Eds., vol. 2. Trondheim Norway: Tapir Acasemic Press, Juni 2007

[12] M. Wieland, U.-P. Käppeler, P. Levi, F. Leymann, and D. Nicklas, "Towards Integration of Uncertain Sensor Data into Context-aware Workflows," in *Tagungsband INFORMATIK 2009 Im Focus das Leben, 39. Jahrestagung der Gesellschaft für Informatik e.V. (GI)*, September 2009

[13] O. Adam, O. Thomas, and G. Martin, "Fuzzy WorkflowsEnhancing Workflow Management with Vagueness," in *EURO/INFORMS Istanbul 2003 Joint International Meeting*, 2003, pp. 6–10.

[14] O. Adam and O. Thomas, "A fuzzy based approach to the improvement of business processes," in *First International Workshop on Business Process Intelligence (BPI05)*, Sep. 2005, pp. 25–35.

[15] S. Urbanski, E. Huber, M. Wieland, F. Leymann, and D. Nicklas, "Perflows for the computers of the 21st century," in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, March 2009, pp. 1 –6.

[16] H. Wolf, K. Herrmann, and K. Rothermel, "Modeling dynamic context awareness for situated workflows," in *OTM 2009 Workshops*, November 2009, pp. 98–107.

[17] S. Buffett and L. Geng, "Bayesian classification of events for task labeling using workflow models," in *Business Process Management Workshops*, Milano, Italy, September 2009, pp. 97–108.

[18] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy Sets and Systems*, vol. 100, no. Supplement 1, pp. 9 – 34, 1999.

[19] J. A. Barnett, "Computational methods for a mathematical theory of evidence," in *IJCAI'81: Proceedings of the 7th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 868–875.

[20] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Prentice Hall, 2002.

[21] Java Universal Network/Graph, *http://jung.sourceforge.net*, 2010.

[22] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *SIGKDD Explor. Newsl.*, vol. 11, no. 1, pp. 10–18, 2009.