Structured Context Prediction: A Generic Approach*

Matthias Meiners, Sonja Zaplata, and Winfried Lamersdorf

Distributed Systems and Information Systems
Computer Science Department, University of Hamburg
[4meiners|zaplata|lamersdorf]@informatik.uni-hamburg.de

Abstract. Context-aware applications and middleware platforms are evolving into major driving factors for pervasive systems. The ability to also make accurate assumptions about future contexts further enables such systems to proactively adapt to upcoming situations. However, the provision of a reusable system component to facilitate the development of such *future-context-aware* applications is still challenging - as it requires to be generic but, at the same time, as efficient and accurate as possible. To address these requirements, this paper presents the approach of *Structured Context Prediction* which constitutes a framework to facilitate the application of existing prediction methods. It allows application developers to integrate domain-specific knowledge by creating a customized prediction model at design time and to select, implement and combine prediction methods for the intended purpose. Feasibility is evaluated by applying a prototype system component to two mobile application scenarios, showing that both high accuracy and efficiency are possible.

1 Introduction

The vision of Ubiquitous Computing fosters the development of smart devices and applications which are able to assist the mobile user while ideally remaining in the background [1]. In consequence, the ability to obtain, to process, to manage and to provide context information describing the user's environment and situation has become one of the most important requirements for such systems. The *prediction of future context* is an important further step which enables devices and applications to also *proactively* support the user [2].

With the ongoing emergence of generic context management systems and middleware support (e.g. [3, 4]), application developers are often able to build context-aware applications on the basis of *generic frameworks*. Supplementary, this paper faces the challenge to offer reusable system support for context prediction (in the following referred to as a *prediction system*) in order to support the development of context-aware applications which should not only consider

* The research leading to these results has received funding from the European Community's Seventh Framework Programme FP7/2007-2013 under grant agreement 215483 (S-Cube).

the current context, but which are also able to derive and use information about future situations – as e.g. illustrated by the following two use cases:

Example 1 (Energy Management). A mobile device uses predictions about its future usage for energy management. If e.g. a user often utilizes his mobile phone in a similar way, the application responsible for energy management can use the prediction system in order to learn about this behavior. Based on predictions, it can make advanced recommendations about the optimal time interval for reloading the battery, save energy in an adjusted way or warn the user in case of critically intensive usage leading to an upcoming uncovered demand for energy.

Example 2 (Service Availability). In mobile ad-hoc networks, the availability of a specific software service often depends on context data such as location or network connections. Thus, service availability can e.g. be predicted by forecasting the position of devices which provide such services and/or of devices which need to consume them. Predictions about service availability can thus be used to improve task assignment by always selecting the most "promising" device [4].

However, such a generic applicability imposes several principal requirements on a prediction system: First, the system should support a preferably wide range of applications and diversity of exploitable contexts [5, p. 11] in order to maximize reusability. Furthermore, there are inter-individual differences between users [6] which can also change continuously [7, p. 77]. Thus, the system has to adapt to the individual user at runtime by learning about the characteristics and regularities which determine the future context (cp. [6] [7, p. 77]).

Besides being generic and adaptive, the prediction system should be able to produce customized predictions for different scenarios in a reliable and satisfying way, i.e. as accurate and efficient as possible. Especially mobile devices often suffer from a lack of resources [8], so that the corresponding requirement for efficiency makes this trade-off even more challenging. Finally, the core motivation to support application developers implies a preferably low effort for them. In summary, the following requirements can be identified:

1. Support for a wide range of applications
2. Support for diverse kinds of context
3. Adaptation to the individual user of the system
4. Accuracy of prediction results
5. Efficiency w.r.t. restricted resources of mobile devices
6. Low effort for application developers

In order to elaborate an approach which fulfills these requirements, the rest of the paper is organized as follows: Section 2 analyzes major existing approaches to context prediction and examines hybrid prediction techniques. Section 3 introduces the approach of *Structured Context Prediction* which partly makes use of such hybrid techniques while integrating domain-specific knowledge in order to achieve accuracy and efficiency. The approach is evaluated in Section 4 using both quantitative evaluation on the basis of Example 2 as well as conceptual evaluation by means of the requirements identified in this section. Finally, Section 5 gives a short summary and an outlook.

2 Background and Related Work

As the basis for every prediction about the future, there must be a sufficient amount of related data collected in the past. In Example 2, the availability of a service and the position of the consuming device are parameters to be considered. Both are called *variables* in this paper and can have different values at different points of time (e.g. position = *at home* and service available = *false*). Such values constitute so-called *historical data*:

Definition 1 (Historical Data). *Historical data of variables V_1, ..., V_n in a time interval with discrete points of time $j \in \mathbb{Z}$ consist of values $v_{i,j}$ for all variables V_i and all points of time j.*

The definition is inspired by time series and stochastic processes as both are very generic concepts. It is used in combination with stochastics in order to express uncertainty about future variable values. Furthermore, different methods may have different types of scales (e.g. nominal or ratio according to statistics). Finally, variables can also be understood as attributes of entities (e.g. the position of the user) establishing the connection to *context* which is defined by Dey and Abowd on the basis of entities [9, p.3-4].

Being on the agenda of context-awareness [5, p. 6], histories provide essential input for prediction. Additionally, this paper takes into account the principle of *inductive learning* (cp. [10, p. 60-61]) which is e.g. used in data mining, machine learning, pattern recognition and statistics. Inductive learning extracts knowledge about characteristics and regularities from observed parts of historical data (*training data*). This is advantageous because storing or repeated processing of complete histories are avoided and thus resources of mobile devices are used more efficiently. The learned knowledge is applied for predictions to infer future context from current and recent context.

The approaches of Mayrhofer [11], Sigg [7] and Petzold [12] are considered to be major contributions towards generic context prediction. Mayrhofer's approach uses an exchangeable prediction method [11, p. 37]. The approach combines context prediction with a preceding extraction of context on a high abstraction level (*high-level-context* [5], e.g. the complex situation *in a meeting*) from context on a low abstraction level (*low-level-context* [5], e.g. the noise level in the current room) [11, p. 5, 33, 62]. Context prediction is simplified by this approach because only few different contexts have to be considered as possible values of only one variable. Thus, a prediction based on high-level-context permits high efficiency. However, only predictions about high-level-context are possible, and Sigg states that such a high-level-context-approach is disadvantageous for accuracy [7, p. 179]. The effort for application developers using Mayrhofer's approach is low [11, p. 128]: Almost no knowledge about the application domain is used [11, p. 128], but its integration is recommended by Mayrhofer as future work [11, p. 132].

Sigg's approach is nearly fully generic, in particular it is not restricted to the prediction of high-level-context [7, p. 92]. On the other hand, the simplification of prediction based on high-level-context is missing. Sigg considers a single prediction method applied at runtime which is exchangeable at design time [7, p.

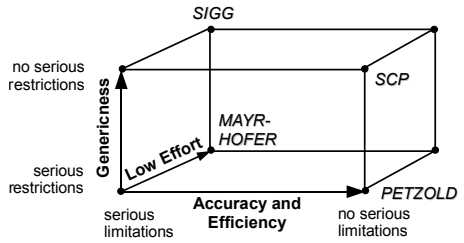

Fig. 1. Characteristics of existing approaches compared to Structured Context Prediction (*SCP*) in view of the requirements identified in Section 1

91], but regardless of the choice of the method, it has to deal with a potentially high number of possible variable/value combinations as well as additional application- and variable-dependent requirements. Such requirements could involve fast predictions of the values of a specific variable, different scale types or the utilization of the type of dependencies among variables [11, p. 66] [7, p. 96]. Mayrhofer, Sigg and Petzold state that there is no universal method fulfilling all possible requirements [11, p. 86, 91] [7, p. 203-204] [12, p. 142]. Thus, it is expected that there will always be serious limitations considering accuracy and efficiency for generic context prediction as long as only a single method is used. The effort for application developers using Sigg's approach is relatively low [7].

Petzold's approach is restricted to the prediction of *primary context* [9], i.e. time, position, identity and activity [12, p. 141], and is therefore not fully generic. In addition to the other approaches, it allows a parallel, hybrid application of multiple methods [12, p. 87] (similarly in [13]). This means that the same prediction task can be assigned to multiple methods in order to better fulfill application- and variable-dependent requirements. The advantages of multiple methods can be combined [12, p. 142], e.g. different specialized methods can be utilized to address different aspects of the prediction. This allows for high accuracy and efficiency. On the other hand, the combination of methods leads to higher effort for application developers who have to select and combine the methods in order to apply the approach to their individual application domains.

Figure 1 summarizes the main characteristics of the presented approaches showing that there is still no approach which is generic enough and offers high accuracy and efficiency at the same time. However, the application of multiple prediction methods is interesting because it offers the possibility to achieve high accuracy and efficiency. Also, the integration of domain-specific knowledge has to be considered because it narrows the prediction task and can simplify achieving high accuracy and efficiency while remaining generic.

Beyond context, Hilario distinguishes different kinds of techniques for a hybrid application of multiple methods [14]. The parallel application of methods which is used by Petzold is called *coprocessing* [14]. In contrast, *chainprocessing* denotes the sequential application of methods so that the prediction result of one method is used as input for another method [14]. Both techniques expose

benefits concerning accuracy and efficiency [14, p. 21-22] [15, p. 1-4]. *Bayesian Networks* offer the possibility to describe a graph-based dependency structure of variables (cp. [16, p. 101, 112-114] [17]). However, such a generic graph-structure is also interesting to describe the connections between methods which take the output of a method as the input of another method (cp. 3.2).

Several methods could be applied for prediction in a hybrid way (e.g. neural networks, regression, decision trees, discriminant functions, markov chains, ARMA and more). However, many of them are not suitable because they require too many resources to be used on mobile devices or do not support adaptive online-learning [2, p. 34] [11, p. 66], i.e. are not able to update already learned knowledge and therefore need an explicit learning phase. The remaining methods come into consideration for the approach presented in the following section.

3 The Approach of Structured Context Prediction

The approach of *Structured Context Prediction (SCP)* realizes a generic prediction system. It is based on fundamental principles derived from the preceding analysis of existing approaches and introduces the new concept of Prediction Nets and an architecture for a corresponding prediction system.

3.1 Fundamental Principles

In order to overcome the remaining conflict between the requirements of genericness, accuracy and efficiency, the proposed prediction system is based on two major principles: In contrast to Mayrhofer who prioritizes unobtrusiveness [11, p. 131], the approach of Structured Context Prediction uses knowledge about the application domain as valuable information which has to be incorporated by the application developers at design time. It is thereby extending Petzold's ideas. The second principle is a hybrid application of multiple, exchangeable prediction methods. Thus, methods which are appropriate to ensure accuracy and efficiency of domain-specific predictions can be selected and combined by the application developers respectively.

The knowledge about the application domain is described as a *prediction model* which specifies the way predictions have to be performed and configures the prediction system. Among other things, it assigns a method to each variable in order to predict its value and interrelates the methods. The method uses the values of other variables as inputs which are again predicted by their own methods or are known (e.g. measured by a sensor). Additionally, an adaptation to the individual user at runtime is achieved by adaptive online-learning as the default learning mechanism. Figure 2 shows how the preceding principles and techniques complement one another in order to apply the generic prediction system to a concrete prediction task.

The integration of domain-specific knowledge can be illustrated by the application responsible for the energy management of a mobile phone (Example 1). The fact that the user is telephoning can be represented as the value of a boolean

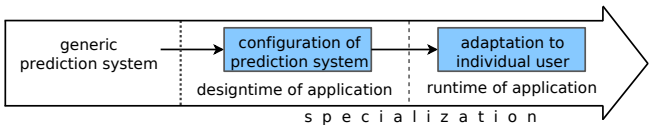

Fig. 2. General methodology of Structured Context Prediction

variable which is predicted by a method using the values of other variables such as the time of day and the position of the user which are again each predicted by their variable's methods. The set of usable methods can be extended by implementing new, possibly application-dependent methods. In consequence, an implementation of a prediction system according to the presented approach constitutes a framework which can be extended by other methods as "plug-ins". So far, a reference configuration of methods is established which is mainly based on *linear regression* and *probability tables*. The notion "probability tables" is used to refer to a method which stores occurrence frequencies of variable/value combinations as knowledge. The properties of the two methods complement one another and are therefore well suited for hybrid application.

From the perspective of an application developer, the whole procedure of using the prediction system consists of two parts: The first part is determined by the development of the prediction model at design time (cp. Section 3.2). The second part is the retrieval of predictions by the respective application at runtime (cp. Section 3.3).

3.2 Prediction Nets

As introduced above, the main part of a prediction model specifies how the methods and respective variables are connected. This part is called *Prediction Net*. A Prediction Net specifies that the value of a variable at a specific point of time is predicted using the values of other variables at the same point of time or earlier (e.g. the position of a user can be predicted by the preceding positions).

Figure 3 shows a simple example of a Prediction Net which is mainly intended to predict the energy consumption of a mobile phone (cp. Example 1). The phone usage at a specific point of time is e.g. predicted by the number of missed calls, the position and the time of day at the same point of time, and the phone usage one and two time steps earlier. Prediction Nets are defined formally as follows:

Definition 2 (Prediction Net). *A Prediction Net is a finite directed graph* $\mathcal{N} = (W, E)$. *The node set* W *is a set of variables* $\{V_1, ..., V_n\}$. *An edge* $V_i \xrightarrow{\Delta} V_{i'} := (V_i, \Delta, V_{i'})$ *in the edge set* $E \subseteq W \times \mathbb{N}_0 \times W$ *expresses that the value of* V_i *at the point of time* $j - \Delta$ *is used as input for the prediction of the value of* $V_{i'}$ *at the point of time* j. *The symbol* Δ *denotes a time offset. The notations* $V_i \rightarrow V_{i'} := V_i \xrightarrow{0} V_{i'}$ *and* $V_i \xrightarrow{1, ..., l} V_{i'} := \bigcup_{k=1}^{l} V_i \xrightarrow{k} V_{i'}$ *are allowed as abbreviations. A Prediction Net contains no cycles of the form* $V_i \xrightarrow{\Delta_1} ... \xrightarrow{\Delta_l} V_i$ *with* $\sum_{k=1}^{l} \Delta_k = 0$.

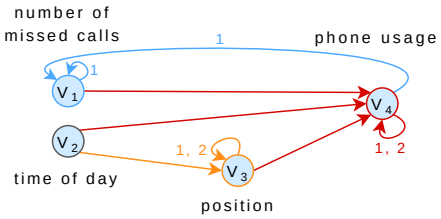

Fig. 3. Prediction Net for Example 1

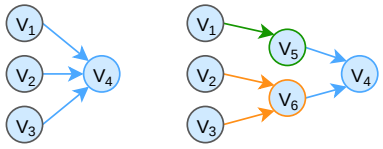

Fig. 4. Prediction Net fragment without (*left*) and with coprocessing (*right*)

Prediction Nets are inspired by Dynamic Bayesian Networks [18] which explicitly take into account the factor *time*. The main difference between Bayesian Networks and Prediction Nets is that Bayesian Networks describe dependencies between variables, and Prediction Nets describe connections between methods which are assigned to the variables. Prediction Nets only allow predictions along the edges of the graph. On the one hand, this makes the design of such a net more complex, but on the other hand it facilitates the predictions – which is advantageous when taking into account the restricted resources of mobile devices. Thus, Prediction Nets are not considered to be a prediction method, but are rather intended as a frame in order to integrate existing methods. It is e.g. possible to use a probability table for a variable with nominal scale type and at the same time regression for another variable with ratio scale type in order to handle a linear dependency in an efficient way with only low storage requirements.

The first step to perform a prediction with a Prediction Net is to generate the relevant part of the respective unfolded Prediction Net. An *unfolded Prediction Net* is a representation of a Prediction Net which represents each variable multiple times, i.e. one variable instance for each point of time. If, e.g., the position of the user (cp. V_3 in Figure 3) should be predicted by the preceding position, the Prediction Net contains the variable V_3 and the edge $V_3 \xrightarrow{1} V_3$. The unfolded Prediction Net is thus determined as ... $\rightarrow V_{3,-2} \rightarrow V_{3,-1} \rightarrow V_{3,0} \rightarrow V_{3,1} \rightarrow V_{3,2} \rightarrow$ ... where ..., -2, -1, 0, 1, 2, ... are points of time (realization of a markov chain).

Prediction Nets are a powerful modeling instrument, e.g. they can be used for coprocessing as shown in Figure 4 (right side). Here, V_5 and V_6 are variables with different methods for the same prediction task. The different results are integrated by the method of V_4 (e.g. by arithmetic mean or majority vote).

3.3 Architecture

Figure 5 shows the architecture proposed by the approach of Structured Context Prediction for a prediction system which can be used as a reusable component in context-aware applications. *Learning* and *prediction* as concurrent processes are mapped to different layers. The two layers are linked by the *knowledge layer* which constitutes a data layer at the bottom of the architecture. The learning layer creates and updates knowledge and the prediction layer uses knowledge for predictions which are performed on demand by default. The architecture

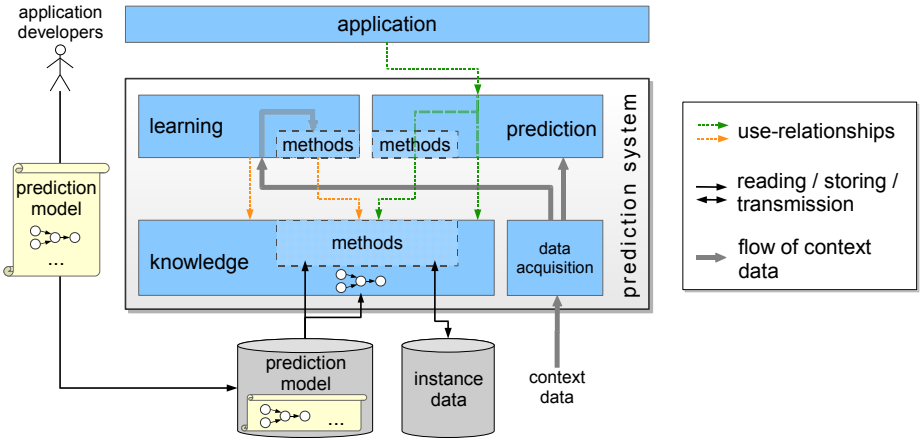

Fig. 5. Architecture of a prediction system according to the SCP approach

permits that the prediction system is located on another device as the application itself (e.g. a powerful server offering location-dependent predictions). The data acquisition can also be performed remotely in order to use sensors of other devices (e.g. GPS). A further possibility of distribution supported by the architecture is to share learned knowledge with other devices.

All of the three mentioned layers contain parts of the methods. Each method possesses its own knowledge (e.g. frequencies of variable values in case of a probability table) and its algorithms for updating the knowledge and predicting the value of the variable associated with the method. However, a method does not have to be aware of the structure of the Prediction Net.

The *knowledge layer* contains knowledge about relationships, characteristics and regularities determining the context. The first part of this knowledge is comprised by the given prediction model which contains the Prediction Net. The second part contains the instance data which is created and updated by adaptive online-learning using the specified methods at runtime in order to adapt to the actual user of the application. In the developed prototype system, the prediction model is created by the application developers as an XML-representation.

The *data acquisition* layer is responsible for acquiring context data. It makes them available by a unique interface and abstracts from the interfaces of physical or logical sensors (cp. e.g. [3]).

The *learning layer* operates concurrently and independently from the application by default. This means that it periodically obtains relevant context data from the data acquisition layer and assigns them to the methods as training data. The methods extract knowledge from the training data by inductive learning. They have to support adaptive online learning unless it is not desired (e.g. in case of a user-independent dependency).

The *prediction layer* makes use of an algorithm which coordinates the methods. This is necessary, because - unlike learning - the prediction normally cannot

be performed by a single method only. For example, if the phone usage should be predicted using the Prediction Net in Figure 3, also the position and the number of missed calls have to be predicted by their respective methods. Accordingly, a prediction initiated by the application begins with the generation of the relevant part of the unfolded Prediction Net. For each relevant point of time for every relevant variable, a prediction unit called *predictor* is created. A predictor obtains input values from its parent predictors and passes them to the method associated with its variable in order to predict the required value at the given point of time. The following algorithm summarizes the prediction of value $v_{i,j}$ of variable V_i at the point of time j:

```
if v_{i,j} already predicted then
  return  v_{i,j}
else
  if v_{i,j} known then
    return  v_{i,j}
  else
    let parent-predictors predict their values
    predict v_{i,j} with own method using these values
    return  v_{i,j}
  end if
end if
```

The algorithm is executed multiple times in order to capture probabilistic behavior. This is inspired by an algorithm called *Stochastic Simulation* which was originally developed for Bayesian Networks [19, p. 189-191] In a *prediction round*, each predictor and its method predict one of the possible values of the corresponding variable at the corresponding point of time. A value should be chosen with high probability only if the probability occurring in reality is also high. The individual prediction results are used to finally obtain a probability distribution. This can either be a distribution of possible values of the variable at a specific point of time in the future, or a distribution of possible points of time in the future when the variable will have a specific value. Additionally to the number of prediction rounds, an alternative *reduced mode* without repeated execution of the algorithm can be chosen. This enables a high scalability in comparison to more usual algorithms for Bayesian Networks which could be adapted for Prediction Nets.

4 Evaluation

A prototype of a generic prediction system according to the approach of Structured Context Prediction has been implemented for the Java Micro Edition™ and was applied to the two application scenarios motivated in example 1 and 2. In particular, the framework is used by the existing *DEMAC*-middleware [4] for the prediction of service availabilities in order to enhance the distribution of mobile business processes. The following subsection presents the experiences which have

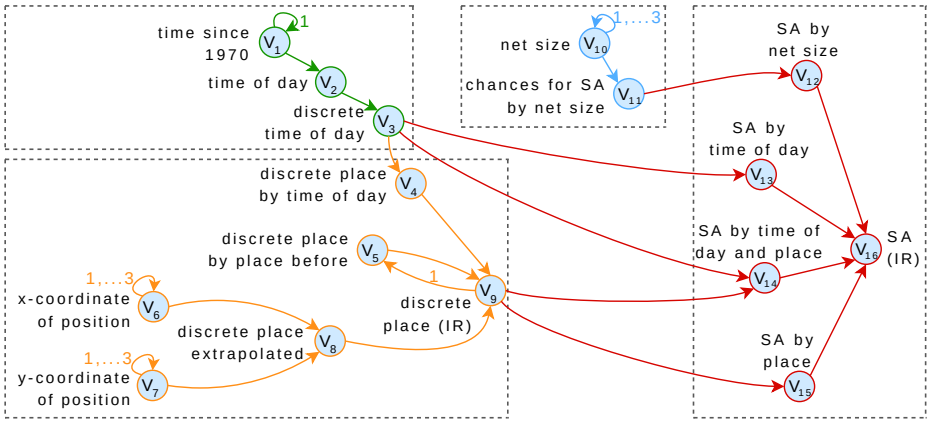

Fig. 6. The Prediction Net for example 2 (*SA* = service availability, *IR* = integrated result of coprocessing)

been made with this second scenario and the developed prototype prediction system. The section concludes with a general conceptual evaluation and discussion.

4.1 Scenario-Based Evaluation

The first part of the evaluation is based on the prediction of service availabilities as motivated by example 2 (cp. Section 1). As a first step, an application-specific prediction model is developed and configured for the prediction of service availabilities for a user's device which is temporarily connected to different ad-hoc networks. In consequence, the *service availability* is assumed to depend on the total *number of devices* within these networks as potential service providers, and, similarly to Figure 3, on the *time of day* and the *position* of the consuming device (e.g. a printer service is available at the location of the user's company for the whole day and an ad-hoc file exchange service is temporarily offered by adjacent mobile devices). Thus, these four variables and the dependencies between them basically constitute the Prediction Net shown in Figure 6.

The mapping of the original four variables of the example to multiple variables in the net results from the selected prediction methods and the use of coprocessing. The *service availability (SA)* at a specific point of time is predicted by using four methods in parallel. Similarly, the *position* at a specific point of time is predicted by using the *time of day* at this point of time as well as the *position* at the preceding point of time (markov chain) and by extrapolating the position. As most of these variables are not numerical, the prediction model uses probability tables as the main prediction method of the reference configuration. In addition, also more specialized methods are used (e.g. such as a method realizing a majority vote and a method for determining the time of day as a periodic variable). Developing an appropriate prediction model for the introduced scenario is not trivial because predictions about arbitrary services

with different characteristics have to be supported. Additionally, because of the resulting network load, service availabilities cannot be measured regularly, so it is e.g. not possible to predict the availability of a service with a markov chain using it's preceding availability.

The example scenario consists of realistic historical data about the behavior of a user and its mobile device spanning an interval of seven days (Monday-Sunday). It contains the *net size*, the *time of day*, the *position* and the *service availability* as values of the corresponding variables at different points of time, representing context data measured by real sensors. For the practical experiment, two services with different behaviors have been chosen: A stationary print service is regularly available when the user is at work. An ad-hoc file exchange service is offered spontaneously by few mobile devices carried by other people in the direct vicinity of the user and is thus only available very unfrequently.

The quantitative evaluation covers accuracy and efficiency. High efficiency means that the ratio of resource consumption and quality of results is appropriate. The efficiency of the prediction system and of the created prediction model is determined by measuring the resource consumption and the accuracy of the prediction results. All results are based on predictions about the availability of a service as a boolean variable at a specific point of time. The evaluation is run on an average notebook (1.5 GHz, Pentium M processor). If appropriate methods (such as in the reference configuration) are used, the memory requirements are bounded and do not significantly increase because the instance data, i.e. the knowledge learned by the prediction methods, is saved instead of measured raw context data. In most cases, the memory consumption is dominated by probability tables. Thus, the upper bound of memory required for the instance data depends on the number of variables connected with the method and the number of their possible values in the Prediction Net. The maximum amount of memory required for the instance data in the example scenario is about 20 KB and the processing time for learning is insignificant (i.e. considerably less than 1% CPU load). The processing time of a prediction depends on the number of variables in the Prediction Net, the number of prediction rounds and the number of time steps in the time interval which is taken into account for the prediction. Theoretical considerations show that – regarding these dependencies – the time complexity is linear. This result is also confirmed by practical experiments regarding the number of time steps in the time interval (cp. Figure 7) and similarly the number of prediction rounds (time consumption ranging from 13 ms for 50 rounds to 117 ms for 500 rounds if a prediction about service availability is requested 60 minutes ahead). Considering the current processing power of smaller mobile devices (e.g. smartphones), the results indicate that also such relatively complex predictions take less than one second and, thus, the resource consumption is relatively well suited even for less powerful mobile devices.

The analysis of accuracy begins with the "empty" prediction system which still has no knowledge learned at runtime. In the course of time, the system learns from the current values of the historical data. Predictions are executed concurrently. The achieved accuracy is determined by comparing the predicted

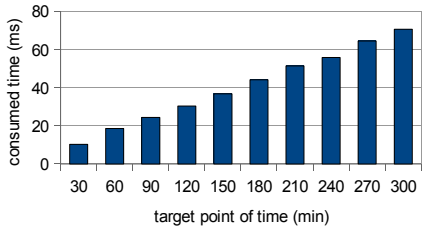

Fig. 7. Prediction of service availability at different future points of time with 70 prediction rounds, every value averaged over 176 predictions

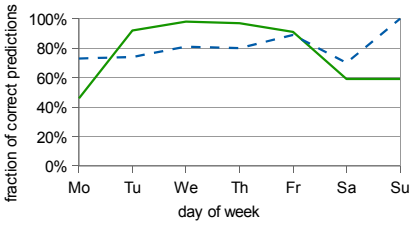

Fig. 8. Correct predictions per day about the availability of a print service (*solid line*) and an ad-hoc file exchange service (*dashed line*)

probability of a service's availability with the actual availability (as boolean value) described in the historical source data. A prediction is considered to be correct if the prediction result states that service availability is probable (resp. unprobable) and the service is actually available (resp. unavailable) in future.

Figure 8 shows the accuracy of predictions about the availability of the two service types at different days. Because the (more simple) ad-hoc file exchange service is often unavailable, this regularity can be learned quickly and predictions about its availability already start with relatively good results, i.e. predicting that the service is not available is correct in most cases. Furthermore, in the following days, the system learns to distinguish the availability of the service and the accuracy slightly increases. Also the prediction results about the availability of the print service improve very quickly. Thus, at Tuesday the system is already able to predict that the printer service is available when the user is at work.

However, the predictions about the ad-hoc file exchange service are still not completely satisfactory, i.e. a trivial prediction approach always predicting that the service will be unavailable would not be significantly worse. Therefore, an enhanced solution could make use of the full potential of coprocessing by automatically preferring the methods with smallest uncertainty arising from the prediction and thus improve the adaption to the individual user. In the case of the ad-hoc file exchange service, e.g. the net size should play a more important role than the time of day and the position which both currently rather disturb predictions.

4.2 Conceptual Evaluation and Discussion

The fundamental principles of the approach of Structured Context Prediction are appropriate to fulfill most of the requirements as identified in Section 1. First, they establish genericness (requirements 1, 2, 3). They enable a configuration of the prediction system which meets application- and variable-dependent demands (e.g. originated by different scale types and dependency types) because methods can be chosen flexibly according to the domain-specific knowledge. Additionally, variables as attributes of entities constitute a generic metamodel in order to

capture the application domain. Thus, many different applications with diverse demands for their domains and possible contexts are supported (requirements 1, 2). An adaptation to the user takes place by adaptive online-learning (requirement 3). The effort for application developers to enable context predictions will be decreased in many cases if an implementation of a prediction system according to the presented approach is used as a reusable system support (requirement 6), because the system coordinates the methods and offers a reference configuration of already implemented methods which can be extended in the future. When all required methods are implemented, the application developers can handle them as black boxes and only have to configure them, combine them and define the data dependencies between them in an abstract way by using the graphical representation of the Prediction Net and the XML-representation of the prediction model. Compared to Mayrhofer's and Sigg's approach, the effort for application developers is still high because domain-specific knowledge is used and has to be incorporated by the application developers. However, this compromise allows for facilitating the prediction task at runtime and for enabling high accuracy and efficiency without limiting the approach of Structured Context Prediction to a special application domain, i.e. keeping it generic.

The possibility to select appropriate methods for the considered application domain is not only a positive aspect in face of genericness, but also in view of accuracy and efficiency (requirements 4, 5). For each variable, the best method to handle the characteristics and regularities determining the value of this variable can be selected. Efficiency can in particular be enhanced if no dependencies are expected between a set of variables. For example, the availability of a service does not necessarily depend on the availability of another service (cp. Example 2). Some dependencies exist, but are known to be instable, i.e. to change from time to time due to unobserved, external influences. Such dependencies are candidates to be ignored so that accuracy and efficiency can be improved. Additionally, it makes sense not to offer and prepare predictions about variables which will never be used. It is e.g. unnecessary to enable predictions about the future position of a device using the information whether a service is available unless it is needed by the application. Finally, the development of an appropriate prediction model can ensure scalability and applicability in the context of heterogeneous devices. For example, the application developers have the possibility to select methods with only a small demand for resources in case the application is targeted to be run on resource-restricted mobile devices.

5 Conclusion

As a further step towards pervasive environments, the approach of Structured Context Prediction realizes a prediction system as a framework with high genericness and potential for high accuracy and efficiency at the same time. So, an approach with a new combination of characteristics in comparison to the approaches analyzed in Section 2 is established (cp. Figure 1). In contrast to analyzed previous approaches which are often restricted to special applications,

e.g. to those which only use high-level-context, the composability of prediction methods and the integration of domain-specific knowledge as proposed here enables support for a wide range of applications. However, there are still some open research tasks - especially in view of the usability of the developed framework. In particular, a reduction of the effort for application developers (e.g. by supporting tools for the development of prediction models) constitutes a major challenge to further facilitate the prediction of future context by context-aware applications.

References

1. Satyanarayanan, M.: Pervasive Computing: Vision and Challenges. IEEE Personal Communications **8**(4) (2001) 10–17
2. Mayrhofer, R.: Context Prediction based on Context Histories: Expected Benefits, Issues and Current State-of-the-Art. In: Proc. of ECHISE '05. (2005) 31–36
3. Salber, D., Dey, A.K., Abowd, G.D.: The Context Toolkit: Aiding the Development of Context-Enabled Applications. In: Proc. of CHI '99, ACM (1999)
4. Zaplata, S., Kunze, C.P., Lamersdorf, W.: Context-based Cooperation in Mobile Business Environments: Managing the Distributed Execution of Mobile Processes. BISE **2009**(4) (2009)
5. Chen, G., Kotz, D.: A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, Dartmouth College (2000)
6. Jameson, A., Wittig, F.: Leveraging Data About Users in General in the Learning of Individual User Models. In: Proc. of. 17th Int. Joint Conf. on Artificial Intelligence. Morgan Kaufmann (2001) 1185–1192
7. Sigg, S.: Development of a Novel Context Prediction Algorithm and Analysis of Context Prediction Schemes. PhD thesis, University of Kassel (2008)
8. Satyanarayanan, M.: Fundamental Challenges in Mobile Computing. In: Proceedings of PODC '96, ACM (1996) 1–7
9. Dey, A.K., Abowd, G.D.: Towards a Better Understanding of Context and Context-Awareness. Technical report, Georgia Institute of Technology (1999)
10. Symeonidis, A.L., Mitkas, P.A.: Agent Intelligence through Data Mining. Springer (2005)
11. Mayrhofer, R.: An Architecture for Context Prediction. PhD thesis, Johannes Kepler University Linz (2004)
12. Petzold, J.: State Predictors for Context Prediction in Ubiquitous Systems. PhD thesis, University of Augsburg (2005) (In German).
13. Petzold, J., et al.: Hybrid Predictors for Next Location Prediction. In: Ubiquitous Intelligence and Computing, Springer (2006) 125–134
14. Hilario, M.: An Overview Of Strategies For Neurosymbolic Integration. In: Connectionist-Symbolic Integration, Lawrence Erlbaum Assoc. (1995) 13–35
15. Dietterich, T.G.: Ensemble Methods in Machine Learning. In: Proceedings of MCS '00, Springer-Verlag (2000) 1–15
16. Borgelt, C., Kruse, R.: Graphical Models - Methods for Data Analysis and Mining. John Wiley & Sons (2002)
17. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1988)
18. Dagum, P., Galper, A., Horvitz, E.: Dynamic network models for forecasting. In: Proceedings of UAI '92, Morgan Kaufmann (1992) 41–48
19. Jensen, F.V.: Bayesian Networks and Decision Graphs. Springer (2001)