Softw Syst Model (2018) 17:973–987
https://doi.org/10.1007/s10270-016-0543-z

SPECIAL SECTION PAPER

Modeling context-aware and intention-aware in-car infotainment systems

Concepts and modeling processes

Daniel Lüddecke[1] · **Christoph Seidl**[2] · **Jens Schneider**[1] · **Ina Schaefer**[2]

Received: 9 February 2016 / Revised: 15 June 2016 / Accepted: 18 June 2016 / Published online: 2 July 2016

Abstract It is fundamental to understand users' intentions to support them when operating a computer system with a dynamically varying set of functions, e.g., within an in-car infotainment system. The system needs to have sufficient information about its own and the user's context to predict those intentions. Although the development of current in-car infotainment systems is already model-based, explicitly gathering and modeling contextual information and user intentions is currently not supported. However, manually creating software that understands the current context and predicts user intentions is complex, error-prone and expensive. Model-based development can help in overcoming these issues. In this paper, we present an approach for modeling a user's intention based on Bayesian networks. We support developers of in-car infotainment systems by providing means to model possible user intentions according to the current context. We further allow modeling of user preferences and show how the modeled intentions may change during run-time as a result of the user's behavior. We demonstrate feasibility of our approach using an industrial case study of an intention-aware in-car infotainment system. Finally, we show how modeling of contextual information and modeling user intentions can be combined by using model transformation.

Keywords Context-aware · Intention-aware · Modeling · Infotainment

Communicated by Dr. Jordi Cabot and Alexander Egyed.

✉ Daniel Lüddecke
daniel.lueddecke@volkswagen.de

Christoph Seidl
c.seidl@tu-braunschweig.de

Jens Schneider
jens.schneider5@volkswagen.de

Ina Schaefer
i.schaefer@tu-braunschweig.de

[1] Volkswagen AG, Letterbox 011/17770, 38440 Wolfsburg, Germany

[2] Technische Universiät Braunschweig, 38106 Braunschweig, Germany

1 Introduction

Computer systems aim to help people by achieving certain goals in their daily routine. However, with the growing complexity of these systems, the number of available functions becomes less comprehensible. One option to counter this effect is to support users in finding suitable functions. By predicting the users' next actions, their *intentions* may be employed to suggest suitable functions. Predicting user intentions may also improve the usability of a computer system when using the system is not the primary task in certain situations. For example, in a car, the primary task for the driver is to operate the vehicle, whereas using the in-car infotainment system to control the music is a secondary task. An in-car infotainment system that is aware of the user's intentions aims to predict a user's next interaction steps to minimize the required interaction between the user and the system. For instance, if the system can predict the next song the driver wants to listen to, or the next navigation destination the driver wants to go to, it may recommend those to the driver. The driver can confirm this recommendation by a single interaction. To the best of our knowledge, there are currently no approaches that attempt to apply model-based software development for intention-aware in-car infotainment systems. In this paper, we present an approach for modeling user intentions by Bayesian networks, which allows the system to react differently depending on the users' intentions in a particular situation.

For the purpose of illustrating challenges for modeling user intentions and to demonstrate our approach for modeling intention-aware systems, we provide a running example of an intention-aware in-car infotainment system that is an excerpt of the industrial example of our case study: the music library of an in-car infotainment system includes information on the genre of every track within the library. Furthermore, the system is capable of playing music of a certain genre. For the running example, we assume there are only three genres within the system: pop, rock and jazz music.

In addition, the car is equipped with a seat occupancy detection, which may be used to determine whether a seat is in use. Hence, the system is able to detect whether the driver is alone or with passengers in the car. Furthermore, the car is equipped with a driver drowsiness detection, which calculates an estimated drowsiness of the driver in percent based on steering wheel movements.

For the running example, we assume that drivers tend to listen to different music genres when they are alone in the car and when there are additional passengers. Furthermore, it is an established theory that people tend to listen to different music genres when they are awake and when they are drowsy (e.g., they are listening to "stimulating music while driving" when they are tired [20]). The exemplary intention-aware in-car infotainment system recommends a specific genre with respect to the user's current situation (being alone or with somebody in the car, being drowsy or awake).

Modeling user intentions with respect to a user's current situation is challenging, as users act differently in the same situation. Hence, the model needs to adapt itself to a specific user. In addition, describing *"the current situation"* can be very complex as a situation is made up by a wide variety of individual facts. Hence, a software designer has to deal with numerous influences that describe the current situation when modeling user intentions.

This paper and its work is mostly based on our results previously presented in [15] and [17]. We present an approach based on Bayesian networks to model user intentions as synthesis of various combined influences. We demonstrate how to apply our approach manually but also provide a procedure to create the model automatically by processing learning data. In addition, we show how to create an integrated process of modeling contextual information as shown in [15] and modeling of user intentions as presented in [17].

This paper is structured as follows: Sect. 2 provides background information on context-aware systems, intention-aware systems and Bayesian networks. Section 3 defines requirements for modeling intention-aware in-car infotainment systems. Section 4 briefly summarizes our previously presented results on modeling context-aware systems with an ontology-based approach. Section 5 presents our approach of modeling user intentions with Bayesian networks to recommend suitable system functions. Section 6 describes how the model may be created automatically by processing learning data. Section 7 demonstrates the feasibility of our approach by applying it to an industrial example of an intention-aware in-car infotainment system. Section 8 presents an optimized process of integrated and flexible modeling contextual information and user intentions. Section 9 discusses related approaches before Sect. 10 concludes the paper with a summary and an outlook to future work.

2 Background

In this section, we provide background information on computer systems that adapt to their users' current situation—*context-aware systems*—and systems that recognize their users' intentions in the current situation—*intention-aware systems*. Furthermore, we present foundations of Bayesian networks, which serve as basis of our approach.

2.1 Context-aware and intention-aware systems

Context-aware systems take the current situation of their users—their *context*[1]—into account when interacting with them. Context-aware systems process low-level data (e.g., sensor data) for a particular situation and react to it by (possibly) altering their behavior [6,26]. Low-level data for an in-car infotainment system can be separated into three main categories according to the origin: *driver*, *car* and *environment* [24,27]. As shown in [15], it is possible to abstractly model different situations using an ontology-based approach by connecting low-level input data to abstract situations (e.g., rain and sun sensor can be used to create the situations *good weather* and *bad weather* during run-time of the system). In this paper, we refer to information that describes the current situation as *contextual information*.

Intention-aware systems aim to reduce the interaction between the computer system and its user by attempting to *predict* which action the user will take next. With this knowledge, the computer system can assist users in achieving their goals, e.g., by automating actions such as starting certain applications. However, there is still a lack of support to model intention-aware in-car infotainment systems.

2.2 Ontology-based modeling of context-aware systems

Ontologies can be used to model real-world concepts and their relations [6,26] among each other. During run-time, the implicit knowledge modeled in ontologies can be made explicit in a process called *reasoning*. A common language

[1] A detailed discussion of the term *context* can be found in [7].

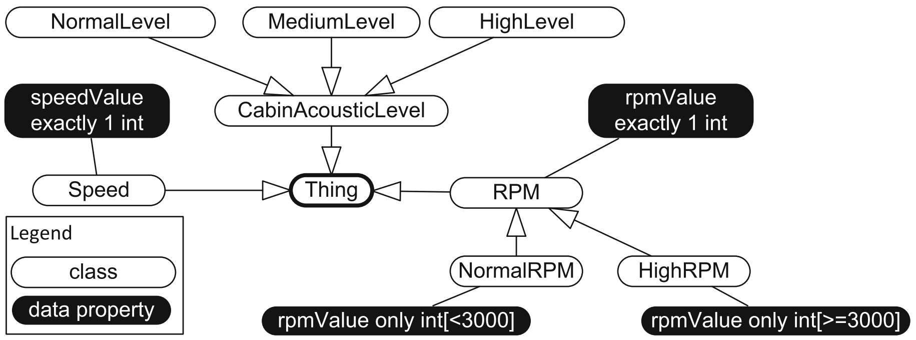


Fig. 1 Example of an OWL ontology for classifying a cabin acoustic level [15]

for ontologies is OWL [19] which was used in our previously presented approach [15] for modeling context-aware systems as well as the rule-based extension called SWRL [12].

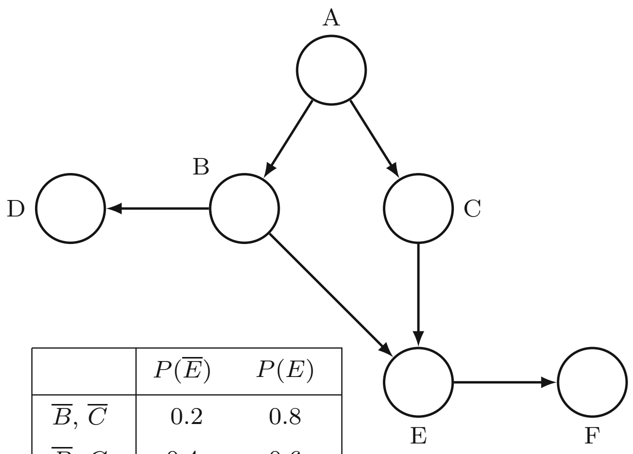
Figure 1 shows a simple exemplary ontology that can be used to classify a cabin acoustic level. Real-world concepts such as the vehicle's speed, the engine's revolutions per minute (RPM) and the cabin's acoustic level are modeled as classes of the ontology. An individual of the class `RPM` is automatically assigned as a member of any of the subclass `NormalRPM` or `HighRPM` according to the value of its property `rpmValue`. SWRL rules can be used to classify an individual as a member of the class `CabinAcousticLevel`. The following SWRL rule classifies a `CabinAcousticLevel` as `HighLevel`:

IF there is an individual of the class `CabinAcoustic Level` (named `c`)
AND there is an individual of the class `HighRPM` (named `r`)
AND there is an individual of the class `Speed` (named `s`)
AND the data property `speedValue` of `s` is greater than 60
THEN `c` belongs to the class `HighLevel`.

Detailed information on how to use ontologies to model contextual information and its relations can be found in [15].

2.3 Bayesian networks

Within this paper, we use the following definition of Bayesian networks:
"Bayes[ian] Networks are directed acyclic graphs in which the nodes represent propositions (or variables), the arcs signify the existence of direct causal influence between the linked propositions, and the strength of these influences are qualified by conditional probabilities [...] [21]." For an easier reading, we use the term *node's value* when we mean the value of a node's variable. We also maintain the more common notation of *edges* instead of *arcs*. In addition to the direct causal influence denoted by edges between nodes, each individual node needs *conditional probabilities* which reveal



	$P(\overline{E})$	$P(E)$
$\overline{B}, \overline{C}$	0.2	0.8
$\overline{B}, C$	0.4	0.6
$B, \overline{C}$	0.1	0.9
B, C	0.7	0.3

Fig. 2 Example of a Bayesian network with a CPT for the node E

how probable a certain value of the node is with respect to the values of its parent nodes. These conditional probabilities are given in *conditional probability tables* (CPTs).

Figure 2 shows a Bayesian network with six exemplary boolean variables, represented by the nodes *A* to *F*, with various dependencies. For example, the value of node *E* depends on the values of node *B* and *C*. The CPT for node *E* denotes probabilities of the variable *E* with respect to the values of variables *B* and *C*.

Bayesian networks allow different basic types of inference support [4]: *predictive support* (also known as *top-down reasoning*) and *diagnostic support* (also known as *bottom-up reasoning*). In this paper, we employ predictive support that allows reasoning about a node's probability by considering the probabilities of the node's parents. For the example in Fig. 2, if given a probability for node *A* and probability distributions for nodes *B*, *C* and *E*, predictive support allows to calculate the probability of the node *F* using the Bayes' Theorem [3]:

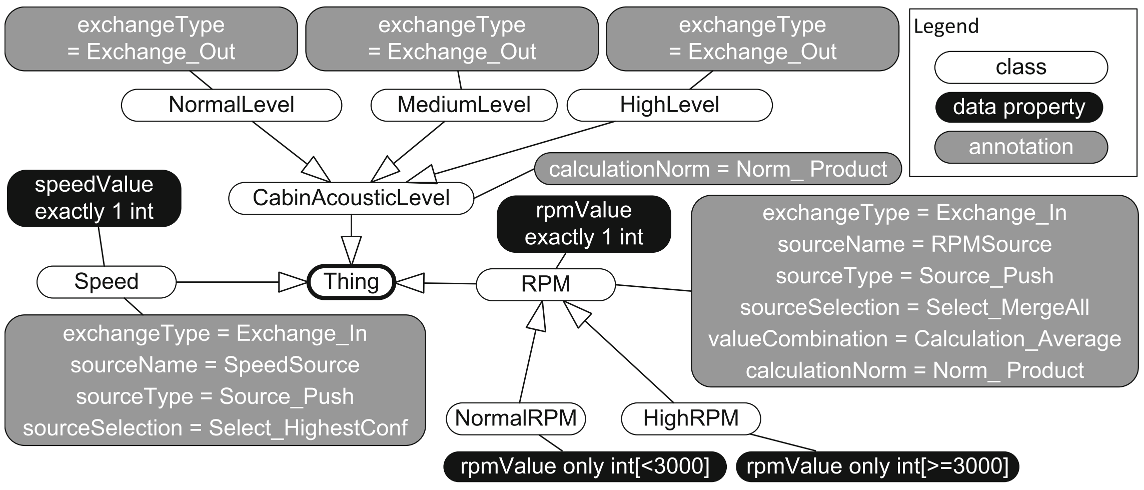

Legend
class
data property
annotation
exchangeType = Exchange_Out
exchangeType = Exchange_Out
exchangeType = Exchange_Out
NormalLevel
MediumLevel
HighLevel
calculationNorm = Norm_ Product
CabinAcousticLevel
speedValue exactly 1 int
rpmValue exactly 1 int
Speed
Thing
RPM
exchangeType = Exchange_In
sourceName = RPMSource
sourceType = Source_Push
sourceSelection = Select_MergeAll
valueCombination = Calculation_Average
calculationNorm = Norm_ Product
exchangeType = Exchange_In
sourceName = SpeedSource
sourceType = Source_Push
sourceSelection = Select_HighestConf
NormalRPM
HighRPM
rpmValue only int[<3000]
rpmValue only int[>=3000]

Fig. 3 Example of classifying a cabin acoustic level supplemented by annotations to enable handling of multiple sources with different reliabilities. [15]

$$P(F|E) = \frac{P(E|F)P(F)}{P(E)}$$

Hence, a Bayesian network supports calculating probabilities for nodes that are not observed during run-time by considering dependencies on this node. This allows inferring knowledge of a fact that is not explicitly but implicitly given by the dependency.

3 Requirements

In this section, we outline five major requirements for intention-aware systems that need to be addressed by an approach for modeling user intentions. The requirements were collected during interviews with experts of the in-car infotainment domain. We illustrate all requirements by using the running example presented in Sect. 1.

3.1 R1: Confidences

A system is required to deliver a confidence for every possible user intention to determine a system function to recommend to users. In addition, confidences are useful to measure how accurate prediction for a certain user intention is with respect to the current input data. Hence, defining or calculating the confidence for every possible user intention should be possible during modeling the system. In the running example, every music genre users may want to listen to needs a confidence, so that the system can decide which music it should play.

3.2 R2: Multiplicity

A system needs to be able to detect that users may have more than one intention at a certain time. Hence, the designer of the in-car infotainment system should be able to express multiple user intentions at a time during modeling of the system. In the running example, users may want to listen to multiple genres, e.g., by mixing rock and pop music.

3.3 R3: Context-awareness

A system is required to take the context into account as intentions of users may depend on the current situation. Hence, an approach for modeling intention-aware in-car infotainment systems should allow modeling context-aware decision making. In the running example, the decision which music the system should play depends on the drowsiness of the driver. From a system's point of view, a drowsy driver is a *contextual information*.

3.4 R4: Adaption

A system needs to be able to adapt during run-time in accordance with individual users as it is not appropriate to model different systems for every possible user. Developers need to be able to model how this adaption to individual users should change the overall system behavior. In the running example, it may occur that individual users of the system want to listen to different music when they are drowsy (e.g., one user prefers rock and another one prefers pop), so that the system needs to adapt to individual users.

3.5 R5: Obliviousness

A system needs to be able to *quickly* forget what it had previously learned if a user's preferences change. This is particularly relevant to avoid disturbing users with recommendations based on *wrong* user intentions. In the running example, a user's taste of music may change over time so that the system has to discard previous knowledge about a user's taste *quickly*.

4 Modeling contextual information

In this section, we briefly summarize our previously presented results of modeling contextual information with an ontology-based approach [15].

Ontologies can help developers to aggregate and abstract certain sensor information to high-level contextual information such as a situation description. Ontologies can also help to capture the great diversity of sensor information (e.g., in terms of update rates). At the same time, ontologies are comprehensible and feature a high degree of extensibility. However, ontologies lack the option to model unreliability of certain values. As presented in [15], we solved this issue by combining ontologies with Fuzzy Logic, which is a common technology for modeling uncertainty in context-aware systems [6]. By adding annotations to classes of an ontology, the implementation of our approach is able to collect live data from multiple sources with different reliabilities during run-time and to reason about the current situation of the driver. In addition, annotations are used to provide supplemental semantics for certain elements of ontologies, e.g., we use annotations to specify from which source certain information can be received during run-time, or how their confidence should be calculated.

Figure 3 shows an example of an ontology that was created to model the cabin acoustic level of a car with respect to the current speed of the car and the engine's RPM. By annotating a class of this ontology with `exchangeType = Exchange_In`, the modeler declares this class as *input class* (cf. classes `Speed` and `RPM`). Other annotations on these classes are used to define the source of this information during run-time (`sourceName`), the type of the source (`sourceType`), and how to deal with multiple sources offering the same information (e.g., a speed information coming from an incremental rotary encoder or the vehicle's GPS sensor). Information that represents a result of the model of contextual information is annotated as *output classes*. This information is used in the next step of modeling user intentions as an input (cf. Sect. 5).

5 Modeling user intentions

In this section, we explain our approach for modeling an in-car infotainment system that is able to recognize its users' intentions based on Bayesian networks. It helps software designers in modeling intentions a user may have in a certain situation by following three steps: (1) modeling input, (2) modeling output and (3) modeling the relations between input and output. Figure 4 illustrates the results of this procedure for the running example. Our approach targets software designers who aim to create an intention-aware in-car infotainment system. However, its concepts are of a general nature so that it may also apply to software systems outside of the domain of in-car infotainment systems.

5.1 Modeling input

The first step in designing an intention-aware in-car infotainment system is to model all relevant input to the system. In this step, the software designer is not required to be aware of the origin of the input data as it is modeled on an abstract level. As described earlier, the input for an intention-aware in-car infotainment system is the current context of driver, vehicle and environment. Hence, the software designer has to know what contextual information is available within the system. However, *detecting* the current context is *not* part of a Bayesian network. The current context is detected in preceding software components, e.g., by using ontologies as described in [15]. Each input to the system (e.g., each contextual information) has to be modeled as a node in the Bayesian network. With respect to the running example, the current context is a *drowsy driver* or a *non-drowsy* driver, and a driver *riding alone* or *riding with somebody*. The *drowsy* or *non-drowsy* driver context is modeled using a Boolean node called *drowsiness*. The context describing whether the driver is currently *riding alone* or *riding with somebody* is modeled using a Boolean node called *aloneInCar*. Figure 4 depicts this contextual information as nodes.

5.2 Modeling output

The second step in designing an intention-aware in-car infotainment system is to model the desired output of the model. As a system should determine which intentions a user may have in certain situations, the output of the model is a set of possible user intentions. Again, each output of the system (e.g., each kind of user intentions) has to be modeled as a node of the Bayesian network. Following the running example, the system contains one kind of user intention—*genre*, with the possible values *pop*, *rock* and *jazz*. Figure 4 illustrates this user intention as node. So far, all possible values have to be modeled within the user intention model. Hence, the modeler should try to cover as a much user intentions as possible and he or she should add values like *Other* to cover unknown values during run-time.

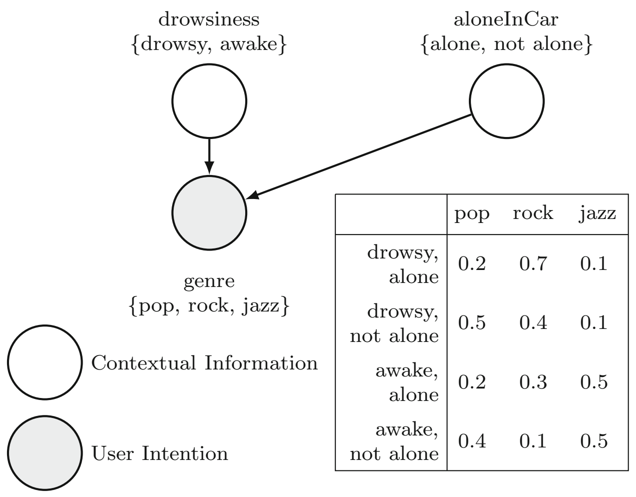
5.3 Modeling relations between input and output

The third step in designing an intention-aware in-car infotainment system is to model which user intention (output) a user may have in which situation (input). This includes

(1) creating edges between nodes to express dependencies between them and (2) defining probability distributions for each intention a user may have for each possible combination of relevant input data. As there is no formal difference between an input and an output node, the software designer is also allowed to create edges between user intention nodes. This allows expressing that one user intention intensifies another user intention.

With respect to the running example, there are dependencies between the nodes *drowsiness* and *genre* as well as between *aloneInCar* and *genre*. As *drowsiness* and *aloneInCar* each have two possible values and genre has three possible values, the software designer needs to specify *12* $(2 \cdot 2 \cdot 3 = 12)$ probabilities. Figure 4 illustrates these probabilities in a CPT. Defining probability distribution can become a very hard task with an increasing amount of input data or an increasing amount of possible values for both input and output. Furthermore, even domain experts do not necessarily know how exactly probabilities are distributed. Section 6 demonstrates how to remedy this problem by learning the probability distribution from collected data.

After defining the input, the output and the relation between the two, the model of an intention-aware in-car infotainment system is complete. At this point, the system of the running example is able to decide which music genre the driver wants to listen to based on whether the driver is drowsy or awake and whether the driver is alone in the car or with somebody.



	pop	rock	jazz
drowsy, alone	0.2	0.7	0.1
drowsy, not alone	0.5	0.4	0.1
awake, alone	0.2	0.3	0.5
awake, not alone	0.4	0.1	0.5

Fig. 4 Bayesian network as model of the running example with manually created structure and probability distribution

6 Learning user intentions

In this section, we outline how structures of the model of an intention-aware in-car infotainment system and probability distributions for the desired user intentions can be learned automatically. We further show how the created model of user intentions may adapt during run-time to maximize the success rate of intention recognition. In addition, we report on experiences with manually modeling user intentions as well as automatically learning structure and probability distributions.

6.1 Learning structure and probability distributions

With a growing set of possible input values for the intention-aware in-car infotainment system, modeling the structure of the system can become a very complex task. As we are using standard Bayesian networks as basis for modeling, we can also use standard *search algorithms* to find the structure of a model such as *K2* [8] or *CB* [25]. However, learning the structure of a Bayesian network requires an appropriate data set that represents dependencies between input data and desired user intentions well enough. Based on these data sets, a search algorithm attempts to find significant dependencies between the different values. As a result, the structure of the Bayesian network is built based on these dependencies. An exemplary data set that automatically generates the structure of the model for the running example is shown in Table 1.

The same data set is used to extract the probability distribution for each desired user intention by employing *estimators* such as *BMAEstimator*.[2] Fig. 5 shows the result of learning structure by using *K2* and probability distributions by *BMAEstimator* for the data of the running example. The learned structure resembles the manually created structure exactly. The probability distribution is slightly different from the manually modeled distribution. However, the order of recommended music genres is the same compared to the manually modeled distribution for each situation.

6.2 Adapting the model during run-time

Intentions of users may change while using the in-car infotainment system and multiple different users may utilize the same system. In consequence, it may be necessary to adapt the intention recognition to the user's behavior. We propose to achieve this by adding data to the learning data set triggered by a user interaction. We distinguish between *positive learning*, which *adds* data to the data set, and *negative learning*, which *removes* data from the data set.

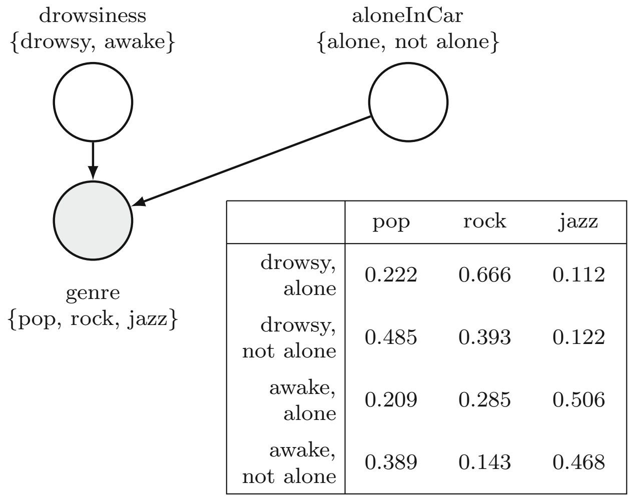
With respect to the running example, positive learning may happen when users select a specific music genre. The selected genre as well as the current situation (*drowsiness* and *aloneInCar*) are then added to the data set. After relearning structure and/or probability distribution, the system is able to take this specific user interaction into account when deciding which music genre the driver wants to listen to. A negative

[2] BMA (Bayes Mean Averaged) estimator as implemented in Weka.

Table 1 Exemplary data set for the running example

drowsiness	aloneInCar	genre
drowsy	alone	pop
drowsy	alone	pop
drowsy	alone	rock
drowsy	alone	rock
drowsy	alone	rock
drowsy	alone	rock
drowsy	alone	rock
drowsy	alone	rock
drowsy	alone	rock
drowsy	alone	jazz
drowsy	notAlone	pop
drowsy	notAlone	pop
drowsy	notAlone	pop
drowsy	notAlone	pop
drowsy	notAlone	pop
drowsy	notAlone	rock
drowsy	notAlone	rock
drowsy	notAlone	rock
drowsy	notAlone	rock
drowsy	notAlone	jazz
awake	alone	pop
awake	alone	pop
awake	alone	rock
awake	alone	rock
awake	alone	rock
awake	alone	jazz
awake	alone	jazz
awake	alone	jazz
awake	alone	jazz
awake	alone	jazz
awake	notAlone	pop
awake	notAlone	pop
awake	notAlone	pop
awake	notAlone	pop
awake	notAlone	rock
awake	notAlone	jazz
awake	notAlone	jazz
awake	notAlone	jazz
awake	notAlone	jazz
awake	notAlone	jazz

learning may happen, when users skip a set of songs that belong to a certain genre. After identifying all the data of the data set that match the current situation and the skipped genre, the identified data (or a subset of it) can be removed from the data set. After relearning structure and/or probability distribution, the system will not recommend the skipped music genre in the same situation.



	pop	rock	jazz
drowsy, alone	0.222	0.666	0.112
drowsy, not alone	0.485	0.393	0.122
awake, alone	0.209	0.285	0.506
awake, not alone	0.389	0.143	0.468

Fig. 5 Bayesian network as model of the running example with learned structure and probability distribution

Modifying the data set by adding and removing data has multiple challenges: adding data over a longer period of time will increase the size of the data set dramatically. Structure and distribution learning will take more time and the effect of adding new data to the data set will decrease. While a new data against a data set with two entries makes up a weight of a 1/3, a new data against a data set with 100 entries only makes up a weight of 1/101. Removing data over a longer period of time may lead to a very small amount of data in the data set. Consequently, the decisions determined by such models may be unreliable. Hence, we recommend to add and remove data from the data set in a balanced way (e.g., just add a new date when removing another at the same time and vise versa), so that the amount of data in the data set will not change dramatically. Another possibility would be to use a ring buffer so that older user interactions would be overwritten by newer ones. This would address the potential issue of permanently increasing memory demand as well as the requirement R5 (Obliviousness). However, our approach did not yet implement such a feature.

6.3 Experiences with learning models

Our experiences show that automatically learning the structure of the model does not always lead to the desired results. Depending on the data set that is used for learning, it may happen that weak dependencies (dependencies that just appear in a small subset of the data set) are not represented in the structure of a manually learned model. However, weak dependencies are important when trying to recognize user intentions. Hence, we recommend to manually create the model's structure or, if it was created automatically, perform a review by domain experts. On the other hand, we have made positive experiences with learning probability distributions after the structure had already been learned. Even adapting and relearning them during run-time seems to be a stable way of adapting to specific users and their intentions in certain situations.

7 Case study

In this section, we demonstrate that our approach based on Bayesian networks is suitable for modeling user intentions in an intention-aware systems. For this purpose, we present a case study on an industrial intention-aware in-car infotainment system called *INIS* (INtention-aware in-car Infotainment System) that was developed within Volkswagen Group Research and is currently in development stage. First, we describe the basic idea behind *INIS* and the user intentions *INIS* can recognize as well as the contextual information the system utilizes. We further show how *INIS* was modeled using our approach. Finally, we assess suitability of our approach with regard to the requirements posed in Sect. 3 based on the results of the case study.

7.1 Description

INIS was created by domain experts of intention-aware in-car infotainment systems at Volkswagen Group Research. They defined that *INIS* is able to recognize three different kinds of user intentions: the music genre the driver wants to listen to, a phone contact the driver wants to call, and a Point of Interest (POI) the driver wants to drive to. To predict these user intentions, the domain experts defined different types of contextual information that describe the current situation of the driver: information on weather, type of the current day (weekend or not), the current time of the day, the kind of passenger in the car (in addition to the driver), the fuel level of the car, the destination of the driver and the car's current position. The domain experts also stated that the decision for a certain music genre is dependent on the weather, the type of the current day, the time of the current day and on the passenger in the car. Furthermore, they stated that the POI used as the driver's destination is dependent on the car's current position, the driving direction and the current fuel level. Finally, they defined that, if the driver makes a phone call, the recipient of the call depends on the number of passengers in the car, the driving direction, the car's current position and the POI used as destination. In addition, the domain experts created a list of rules that express user behavior tendencies:

- Users tend to listen to pop music on the weekend.
- Users tend to listen to rock music on a weekday's morning.
- Users tend to search for a gas station when they are driving home, are near home and the fuel level is low.
- Users tend to search for a parking possibility when they are driving to work and are near work.
- Users tend to call their first phone favorite when they are alone in the car, driving home and are not looking for a gas station.
- Users tend to call their second phone favorite when children are in the car and the car is near the workplace.

As not all of the contextual information used by *INIS* is based on simple sensor values, it is aggregated in a upstream software component that was modeled using ontologies as described in [15], called *context aggregation* in the following. Hence, it is *not* the focus of the case study to determine how the system detects this contextual information (i.e., how the system gathers information about the passenger). Instead, the case study will show that if there is knowledge about dependencies between contextual information and a certain user intention, our approach is suitable to model this knowledge to be used in an intention-aware in-car infotainment system.

We use Java Bayes[3] to manually create the structure of our models. We further use Weka[4] to automatically learn probability distributions. In addition, we have implemented several software components to integrate Bayesian networks in our existing model-based software architecture and to get predictions for user intentions during run-time.

7.2 Methods

To show that *INIS* can be modeled using our approach, we created a Bayesian network that includes all user intentions that *INIS* can recognize as well as all available contextual information. The three kinds of user intentions were modeled by creating three user intention nodes in the Bayesian network:

1. `Music Genre` can take a different value for each supported music genre: `Blues`, `Classical`, `Country`, `Disco`, `HipHop`, `Jazz`, `Metal`, `Pop`, `Rock`, `Reggae`.
2. `POI Search` can take three different values representing points of interest the driver wants to drive to: `Parking`, `GasStation`, `None`.
3. `Calling Contact` can take four different values representing three favorite contacts[5] the driver saved in the in-car infotainment system and the fact that the driver does not want to call any contact: `Favorite1`, `Favorite2`, `Favorite3`, `None`.

[3] http://www.cs.cmu.edu/~javabayes/Home/.

[4] http://www.cs.waikato.ac.nz/ml/weka/.

[5] As our approach currently requires the modeler to know possible user intentions during modeling time, the system is not able to predict a probability for a certain telephone number but for a certain contact from favorites.

Input to *INIS* is contextual information that is created during context aggregation. Hence, contextual information is not just low-level sensor data but a high-level situation description. Contextual information was modeled by creating the following seven contextual information nodes:

1. `Weather` can take the values `Good` and `Bad`. These values are set by context aggregation during run-time. Our model is oblivious to which low-level data is used to create this high-level data. However, it is most likely based on values for *rain intensity*, *sun intensity* or *temperature*.
2. `Day Type` can take the values `Weekday` and `Weekend` according to the current day of the week.
3. `Day Time` can take the values `Morning`, `Noon`, `Afternoon` and `Night` with respect to the current time of the day. Again, which time will actually create which value is not known to the model at this point.
4. `Passenger` can take the values `None`, `Children`, `Friends` and `Other`.
5. `Driving To` can take the values `Home`, `Work` and `Other`.
6. `Fuel Level` can take the values `Low`, `Medium` and `High`.
7. `Near To` can take the values `Home`, `Work` and `Other`.

In addition to modeling contextual information and user intentions, we modeled dependencies between those two by creating connections between corresponding nodes. The user intention `Genre` is dependent on the contextual information `Weather`, `Day Type`, `Day Time` and `Passenger`. The user intention `POI Search` is dependent on the contextual information `Driving To`, `Fuel Level` and `Near To`. Finally, the user intention `Calling Contact` is dependent on the contextual information `Passenger`, `Driving To`, `Near To` and the user intention `POI Search`. In order to create the probability distributions of the user intention nodes, we employed the previously modeled structure and a dataset[6] representing the rules of user behavior tendencies for all user intention nodes. Using the *BMAEstimator*, we created our final model of *INIS*. During run-time, we calculate probabilities for the defined user intentions. For the purpose of retrieving probabilities, we created several software components to execute the following tasks:

1. Collect contextual information at the software component doing context aggregation.
2. Set collected values in the Bayesian network as *observed values*.
3. Query the nodes of interest at the Bayesian network to get probability distributions for the appropriate user intentions.

[6] The dataset was stored using the *Attribute-Relation File Format (ARFF)* (http://www.cs.waikato.ac.nz/ml/weka/arff.html).

7.3 Results and discussion

Figure 6 shows the structure of the model resulting from the case study. Dependencies between contextual information and user intentions of interest could be modeled using the implementation of our approach based on standard tools. As probability distributions for a node grow rapidly with an increasing amount of parent nodes and possible node values, we have to omit the CPTs for the user intention nodes. With this model, it is possible to determine user intentions within *INIS*. In order to assess suitability of our approach, we examine the results of the case study with regard to the requirements posed in Sect. 3:

7.3.1 R1: Confidences

Confidences for user intentions are guaranteed as we are using Bayesian networks. The system is able to calculate confidences for each and every user intention based on the probability distributions given or learned for all user intentions. However, our case study shows that defining probability distributions manually is a process that does not scale very well with a growing set of input data. For example, manually creating CPTs for the user intention of the case study would have $2 \cdot 2 \cdot 4 \cdot 4 \cdot 10 = 640$ cells. Hence, we strongly recommend learning probability distributions by using estimators as described in Sect. 6.

7.3.2 R2: Multiplicity

During run-time, confidence values are added to every modeled user intention. For example, during run-time, the system's output may be: the user wants to listen to rock to 50 % and to pop to 45 % (all other genres may share the remaining percentage). Having this distribution, the system may create a playlist as a mixture of both genres. Hence, multiplicity can be achieved by modeling multiple user intentions.

7.3.3 R3: Context-awareness

User intentions can be modeled to be dependent on the current situation of driver, vehicle and environment. Contextual data can be used as input to decide how probable a certain user intention is in the current situation. In addition, we have shown in previous work how to enable software designers of an intention-aware in-car infotainment system to abstract from raw sensor data by using aggregated context data that was modeled using ontologies [15].

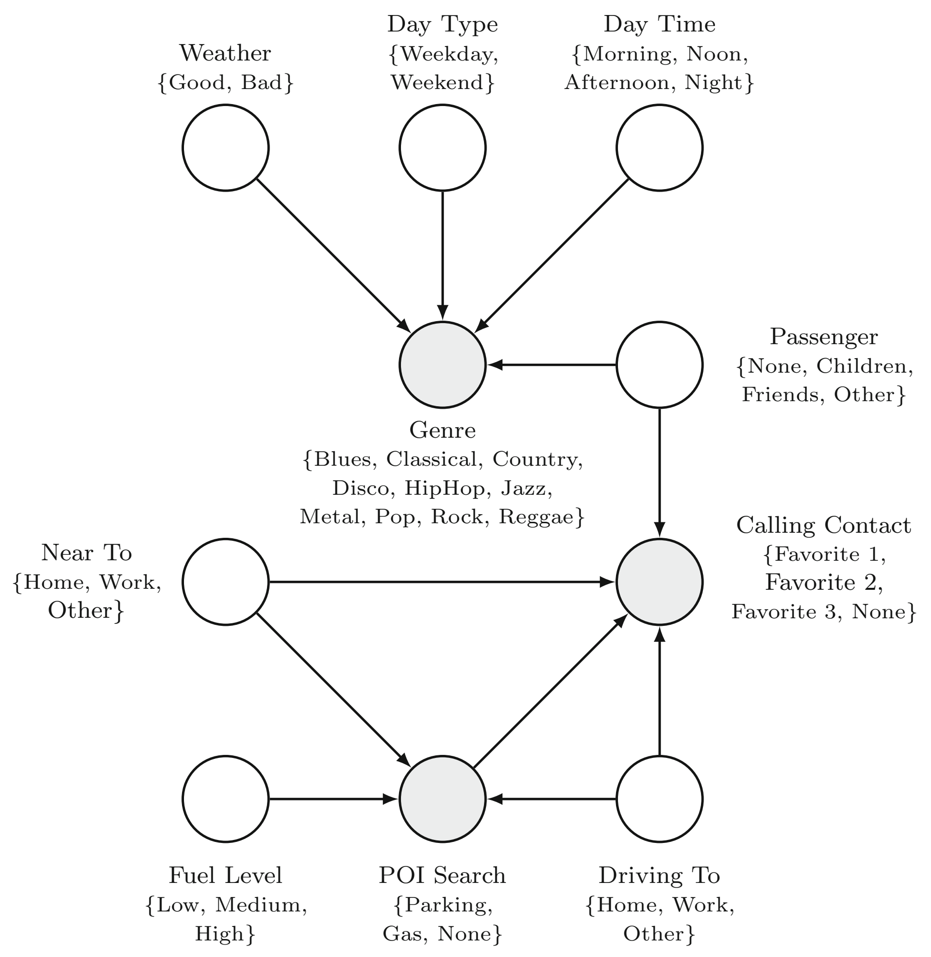


Fig. 6 Structure of a Bayesian network as model of *INIS*

7.3.4 R4: Adaption

Adapting the model during run-time can be achieved by relearning the model's parameters. Relearning can take place at two different layers: first, relearning the structure of the model relearns dependencies between input (contextual information) and output (user intentions). Second, learning probability distributions of user intentions relearns user intentions for a certain situation. Our experiences show that relearning the latter is a good opportunity to adapt the predefined intention-aware system to a user's individual behavior. However, learning the structure of the model may lead to unintended behavior of the system. Hence, we recommend to only adapt the probability distribution and to keep the model structure during run-time.

7.3.5 R5: Obliviousness

Forgetting previously learned user intentions if the user's behavior changes is possible by adapting the probability distributions of the present user intentions. However, our experiences show that recognizing a change of the user's behavior often takes too long and, hence, confuses or even frustrates users because the system may disturb them with *wrong* user intentions. Hence, obliviousness cannot yet be achieved completely using our approach.

With respect to our posed requirements, our approach of modeling intention-aware in-car infotainment systems can be assessed as suitable. However, some requirements are satisfied only partially and need to be addressed further in future work. In particular, adapting the structure of the model during relearning needs additional investigations (R4). Furthermore, obliviousness (R5) needs more attention in future work.

With the case study, we have shown that modeling an intention-aware in-car infotainment system, which is defined by domain experts in a non-formal way, using Bayesian networks is possible. We further expect definite benefits when adapting the Bayesian network during run-time with respect to the user's interaction. For example, if users start to listen to Jazz music at night on multiple occasions, our system is able to learn this behavior and recommend this music to the user in a similar situation. However, we do not yet have experiences with such long-term usage of our system. For example, we do not yet know how often users have to listen to a certain genre at a certain time until the system starts recognizing it as an explicit intention so that the genre can be recommended to users.

8 Modeling processes

In this section, we have a holistic view of our two approaches of modeling contextual information using ontologies [15] which are subsequently used to model user intentions with Bayesian networks [17]. We identified drawbacks during our testing and operation of intention-aware in-car infotainment systems such as *INIS* and others, with regard to the integration of modeling contextual information and the modeling of user intentions, and to the flexibility of the used modeling technology. Although the current implementation of those systems is running correctly, stable, and with sufficient performance [16], we experienced drawbacks stemming from the process of modeling intention-aware systems. This section explains the three biggest drawbacks in detail and shows how to overcome these drawbacks by modeling the system as a whole.

8.1 Drawback I: no integrated modeling process

Our two approaches of modeling contextual information and of modeling user intentions share a certain set of information: contextual information, which is modeled using ontologies, and which is represented by input nodes of a Bayesian network and the thereby modeled user intentions. However, so far, there is no connection between modeling contextual information and modeling user intentions. People modeling user intentions currently need to know which contextual information is available within the system which can be used during the modeling of user intentions. This process can be extensive (e.g., the modeler has to look up the name for a specific contextual information if her or she does not know how a contextual information is named) and even error-prone (e.g., if the modeler fails to write the name of a needed contextual information correctly).

8.2 Drawback II: inflexible modeling of contextual information

Modeling contextual information and its relations is done by creating ontologies that translate real-world concepts and relations into a machine-readable format. However, we identify the drawback that our presented approach [15] is inflexible with respect to exchanging the knowledge store technology for contextual information. Hence, it is very reasonable to us to support other knowledge storage technologies than ontologies.

8.3 Drawback III: inflexible modeling of user intentions

Modeling user intentions with Bayesian networks as presented in [17] works well for certain use cases. However, we were confronted with user behavior that does not seem to be well-predictable by using Bayesian Networks, but by using different pattern-detecting technologies (e.g., Neural Networks, Decision Trees, or a technology mix). Hence, we identify the drawback of our presented approach of not being able to deal with other technologies than Bayesian networks. In addition, our approach does not support multiple technologies in parallel during run-time, which is necessary to support different use cases at the same time.

8.4 Solution: flexible and integrated modeling process

The following section presents our solution to all of the drawbacks mentioned before. Figure 7 shows our proposal of an integrative and flexible process of modeling context and intention-aware systems to decouple the two main tasks of modeling contextual information and user intentions.[7]

We propose to use an intermediate model called *System Model* and use model transformation to transform from and to existing models. Ontologies or any other knowledge storage technology can be transformed to templates of system models that contain all possible contextual information (e.g., classes that were defined as an *output class* in an ontology). It is also important to allow using more than one model as input for this step to cover the case of a model being split into multiple sub-models for the purpose of parallel modeling by more than one modeler.

Next, the modeler refines this system model template by defining which modeling technologies of user intentions will be used (e.g., Bayesian networks or neural networks) and which user intentions will be detected by these technologies. Also, the modeler adds which contextual information is needed for a certain user intention detected by a certain modeling technology. We call this step *mapping* of contextual information, user intentions and used technology. As a result, the modeler receives the final system model. This final system model can be transformed to templates of each and every user intention modeling technology. For example, the final system model can be transformed to a Bayesian network that contains nodes for every user intention that was defined in the system model to be detected by the Bayesian network, and nodes for each contextual information that is needed to detect these user intentions. Templates of user intention models can then be used by modelers to create the final user intention

[7] A rectangular box represents a task. A rectangular box with a bent upper right corner represents a document. A rectangular box with a bent upper right corner surrounded by a dotted line represents multiple documents. Arrows represent the work flow.

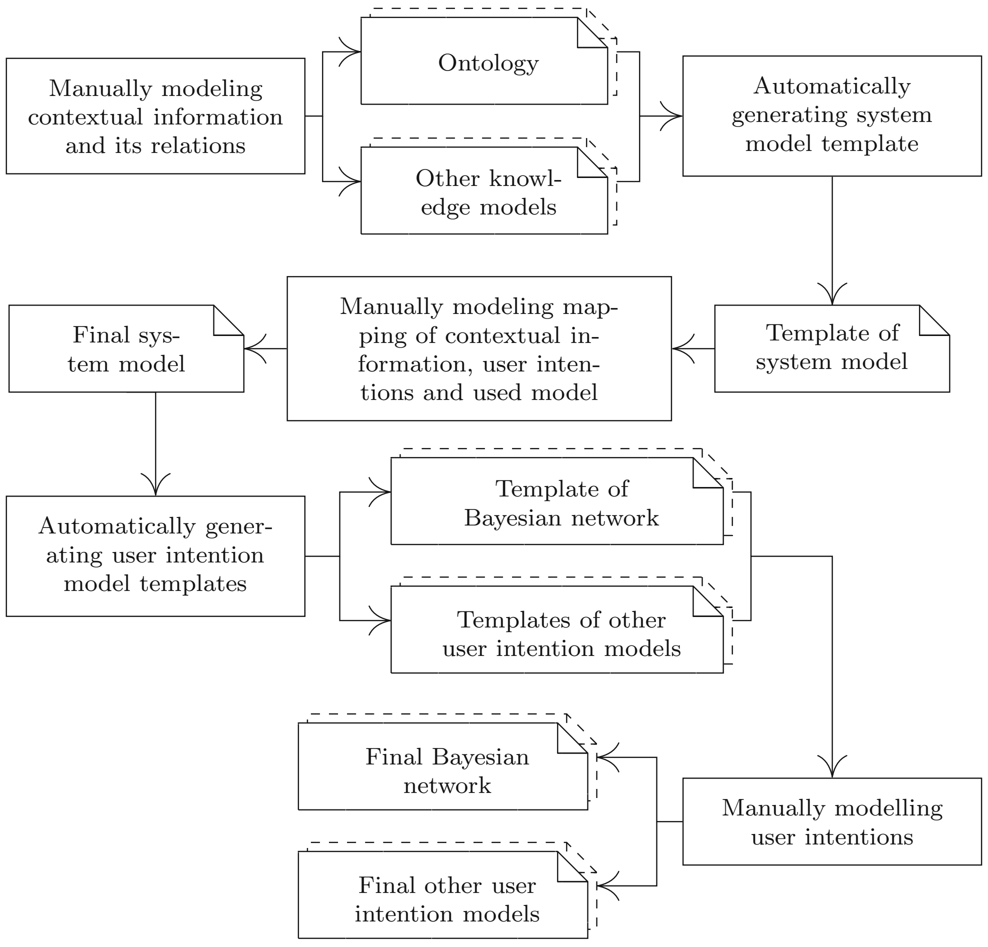

Manually modeling contextual information and its relations
Ontology
Other knowledge models
Automatically generating system model template
Template of system model
Manually modeling mapping of contextual information, user intentions and used model
Final system model
Automatically generating user intention model templates
Template of Bayesian network
Templates of other user intention models
Manually modelling user intentions
Final Bayesian network
Final other user intention models

Fig. 7 Process of integrative modeling of contextual information and user intentions

model (e.g., the final Bayesian network). An alternative solution would be to directly transform to a template of a Bayesian network from ontologies and other knowledge models. However, if there was any change in how the Bayesian network template should be structured (e.g., how nodes are named), it would involve changes in each and every model transformer that transforms Bayesian network templates from any knowledge modeling technology, making the adaption error-prone. Hence, adding a layer between the model of knowledge and the model of user intentions decouples this dependency. If there was any change in how Bayesian network templates should look like, only the transformer that transforms the system model to the Bayesian network template needs to be adapted.

8.5 Tool chain

As mentioned before, we use model transformation for automatically creating templates of the system model and user intention model. These transformations are currently done by tools we developed during establishing the integrative modeling process. This was the best possibility for researching and developing the process because we were able to quickly adapt the tools to our needs. However, we recommend to integrate this process in tools that are used to model future in-car infotainment systems. For example, the transformation of models could be done by using the *Model-to-Model Transformation*[8] SDK which is part of the *Eclipse Modeling Project*.[9]

8.6 The system model

We have used XML as language for the system model as it offers both human and machine readability. However, we think that this decision might be revised in productive systems. Nevertheless, the central role of the system model remains a very important part of our proposed process.

[8] Model-to-Model Transformation Web site: https://projects.eclipse.org/projects/modeling.mmt.

[9] Eclipse Modeling Project Website: https://projects.eclipse.org/projects/modeling.

First, the system model is very crucial for the flexibility of the modeling process. The system model decouples knowledge models, such as ontologies, from user intention models, such as Bayesian networks. This decoupling ensures that changes in any user intention model just requires changes in the model transformer that transforms the system model to a user intention model. Without the system model (e.g., if there were multiple model transformers for any permutation of knowledge and user intention models), such changes would involve changes in multiple model transformers and, hence, might result in a higher error-proneness.

Second, the system model defines the need for a certain contextual information with respect to a certain user intention. This helps to improve the usability of user intention models. Without the system model, user intention models would include each and every contextual information. This would dramatically increase the size of these models, which can have a negative influence on both readability of the model and performance during run-time.

9 Related work

Recognizing a user's intentions is also used in robotics to train robots based on human behavior [13]. Robots also need to predict human intentions when human beings and robots have to share an environment to avoid collisions [2] or when robots help elderly or disabled people as an *intelligent wheelchair* [9]. A popular approach for modeling and learning user intentions in robotics are *Hidden Markov Models* (HMMs) [18]. They are well suited for representing sequences of actions. As we need to know what intention a user may have with respect to only the *current* situation, we decided against HMMs. However, our latest experiences show that temporal dependencies between certain facts may also be of interest when reasoning about user intentions.

Very recently, the creation of personal assistants aims to support their users with the right information at the right time. Therefore, they need to understand what information is of interest for a certain user in a certain situation. They try to achieve this by *modeling users* [11]. There are also approaches which aim to model users in social media systems [28].

In an automotive context, predicting user intentions is needed for advanced Driver Assistance Systems (DASs) [5, 14,23]. An overview of DASs that predict human behavior or intentions can be found in [10]. The prediction of user intentions is further used to create an intelligent in-car infotainment system that "proactively automate(s) complex interaction tasks that are supposed to happen anyway" [22]. Also, Ablaßmeier et al. presented an agent-based approach that also uses Bayesian networks to predict the user's next interactions [1].

10 Conclusion

Current in-car infotainment systems are created using model-based development. However, they do not yet take user intentions into account. In this paper, we presented an approach that allows a model-based development of an intention-aware in-car infotainment system. Our approach is based on Bayesian networks and allows modeling dependencies between contextual information, which describes the current situation of the driver and possible user intentions. We have shown that learning probability distribution of nodes in the Bayesian network during run-time can help in adapting the system's behavior to individual users.

As future work, we will investigate approaches to improve obliviousness of the Bayesian network to reduce wrong user intention recognition. Furthermore, we will survey whether adapting the structure of Bayesian networks during run-time is necessary to improve user intention prediction. We also plan on evaluating and refining our holistic approach of modeling contextual information and user intentions and to supply this process with sufficient tool support.

Finally, future work needs to evaluate influences such systems may have on the overall user experience and how an intention-aware in-car infotainment system can still offer possibilities to reach functions their users have never used. By modeling contextual information and user intentions, the work presented in this paper enables researchers, user experience designers and developers to rapidly create and adjust intention-aware systems to do extensive user studies to evaluate how good the prediction of user intentions can be.

References

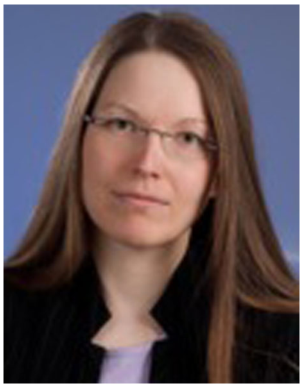
1. Ablaßmeier, M., Poitschke, T., Reifinger, S., Rigoll, G.: Context-aware information agents for the automotive domain using bayesian networks. In: Smith, M.J., Salvendy, G. (eds.) Human Interface and the Management of Information. Methods, Techniques and Tools in Information Design. Lecture Notes in Computer Science, vol. 4557, pp. 561–570. Springer, Berlin (2007)
2. Bandyopadhyay, T., Won, K.S., Frazzoli, E., Hsu, D., Lee, W.S., Rus, D.: Intention-aware motion planning. In: Frazzoli, E., Lozano-Perez, T., Roy, N., Rus, D. (eds.) Algorithmic Foundations of Robotics X, Springer Tracts in Advanced Robotics, vol. 86, pp. 475–491. Springer, Berlin (2013)
3. Bayes, Price: An essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, f. r. s. communicated by mr. price, in a letter to john canton, a. m. f. r. s. Philos. Trans. Royal Soc. Lond. **53**, 370–418 (1763)
4. Ben-Gal, I.: Bayesian networks. In: Ruggeri, F., Kenett, R.S., Faltin, F.W. (eds.) Encyclopedia of Statistics in Quality and Reliability. Wiley, Chichester (2008)
5. Berndt, H., Emmert, J., Dietmayer, K.: Continuous driver intention recognition with hidden markov models. In: 11th International IEEE Conference on Intelligent Transportation Systems (ITSC), pp. 1189–1194. (2008)

6. Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D.: A survey of context modelling and reasoning techniques. Pervasive Mob. Comput. **6**(2), 161–180 (2010)
7. Chen, G., Kotz, D.: A survey of context-aware mobile computing research. Technical Report TR2000-381, Dartmouth College, Computer Science, Hanover (2000)
8. Cooper, G., Herskovits, E.: A bayesian method for the induction of probabilistic networks from data. Mach. Learn. **9**(4), 309–347 (1992)
9. Dirk Vanhooydonck, Eric Demeester, Marnix Nuttin, Hendrik Van Brussel: Shared control for intelligent wheelchairs: an implicit estimation of the user intention. In: Proceedings of the ASER ’03 1st International Workshop on Advances in Service Robotics, pp. 13–15. (2003)
10. Doshi, A., Trivedi, M.M.: Tactical driver behavior prediction and intent inference: a review. In: 14th International IEEE Conference on Intelligent Transportation Systems—(ITSC 2011), pp. 1892–1897. (2011)
11. Guha, R., Gupta, V., Raghunathan, V., Srikant, R.: User modeling for a personal assistant. In: Proceedings of the Eighth ACM International Conference on Web Search and Data Mining. WSDM ’15, pp. 275–284. ACM, New York (2015)
12. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML (2004). http://www.w3.org/Submission/SWRL
13. Iba, S., Paredis, C., Khosla, P.K.: Intention aware interactive multimodal robot programming. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3479–3484. (2003)
14. Kuge, N., Yamamura, T., Shimoyama, O., Liu, A.: A Driver Behavior Recognition Method Based on a Driver Model Framework. SAE International, Warrendale (2000)
15. Lüddecke, D., Bergmann, N., Schaefer, I.: Ontology-based modeling of context-aware systems. In: Proceedings of the 17th International Conference on Model Driven Engineering Languages and Systems. Lecture Notes in Computer Science, vol. 8767, pp. 484–500. Springer International Publishing, Cham (2014)
16. Lüddecke, D., Seidl, C., Schaefer, I.: Efficient ontology-based modeling of context-aware in-car infotainment systems: Benchmark infrastructure and design guidelines. In: Proceedings of the International Workshop on Modelling in Automotive Software Engineering, CEUR Workshop Proceedings vol. 1487, pp. 23–32. (2015)
17. Lüddecke, D., Seidl, C., Schneider, J., Schaefer, I.: Modeling user intentions for in-car infotainment systems using bayesian networks. In: ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS), pp. 378–385. (2015)
18. Markov, A.A.: An example of statistical investigation of the text eugene onegin concerning the connection of samples in chains. In: Proceedings of the Academy of Science, p. 153. St. Petersburg (1913)
19. Motik, B., Patel-Schneider, P.F., Parsia, B.: OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax, 2nd edn. (2012). http://www.w3.org/TR/owl-syntax
20. Nguyen, L.T., Jauregui, B., Dinges, D.F.: Changing Behaviors to Prevent Drowsy Driving and Promote Traffic Safety: Review of Proven, Promising, and Unproven Techniques. AAA Foundation for Traffic Safety, Pennsylvania (1998)
21. Pearl, J.: Bayesian networks: A model of self-activated memory for evidential reasoning, 1985. In: Proceedings of the 7th Conference of the Cognitive Science Society, pp. 329–334. University of California, Irvine (1985)
22. Rodriguez Garzon, S.: Intelligent in-car-infotainment system: a prototypical implementation. In: 8th International Conference on Intelligent Environments (IE), pp. 371–374. (2012)
23. Salvucci, D.D.: Inferring driver intent: a case study in lane-change detection. Proc. Hum. Factors Ergon. Soc. Annu. Meet. **48**(19), 2228–2231 (2004)
24. Schäuffele, J., Zurawka, T.: Automotive Software Engineering, 5th edn. Springer, Berlin (2013)
25. Singh, M., Valtorta, M.: Construction of bayesian network structures from data: a brief survey and an efficient algorithm. Int. J. Approx. Reason. **12**(2), 111–131 (1995)
26. Strang, T., Linnhoff-Popien, C.: a context modeling survey. In: Proceedings of the 6th International Conference on Ubiquitous Computing. Workshop on Advanced Context Modelling, Reasoning and Management. Lecture Notes in Computer Science (LNCS), vol. 3205. Springer, (2004)
27. Winner, H., Hakuli, S., Wolf, G.: Handbuch Fahrerassistenzsysteme: Grundlagen, Komponenten und Systeme für aktive Sicherheit und Komfort, 1st edn. Vieweg+Teubner, Wiesbaden (2009)
28. Yin, H., Cui, B., Chen, L., Hu, Z., Zhou, X.: Dynamic user modeling in social media systems. ACM Trans. Inf. Syst. **33**(3), 10:1–10:44 (2015)

Daniel Lüddecke is a researcher at Volkswagen Group Research in Wolfsburg, Germany. He got his master’s degree in computer science from Otto-von-Guericke-Universität Magdeburg, Germany. In his master’s graduation work, he focused on how to extract features and feature models from a given set of implementation artifacts in a semi-automated process. His current research at Volkswagen Group Research revolves around modeling contextual information and user intentions to predict a user’s next interaction with the in-car infotainment systems.

Christoph Seidl is a postdoctoral research fellow at the Institute of Software Engineering and Automotive Informatics of Technische Universität Braunschweig, Germany. He received his Ph.D. in Computer Science in 2016 from Technische Universität Dresden, Germany. For his dissertation on managing variability in space and time in software families, he received the SAP award for the most industry-relevant Ph.D. thesis of the year. His research revolves around variability, software evolution, model-based software development and domain-specific languages.

Jens Schneider is a researcher at Volkswagen Group Research in Wolfsburg, Germany. He got his master's degree in computer science from Otto-von-Guericke-Universität Magdeburg, Germany. In his master's graduation work, he focused on machine learning methods for the prediction of individual user preferences to improve in-car speech interaction. His current research at Volkswagen Group Research revolves around artificial intelligence, machine learning and speech interaction for in-car infotainment systems.



Ina Schaefer is chair holder at the Institute of Software Engineering and Automotive Informatics of the Technische Universität Braunschweig. She received her Ph.D. in computer science by the TU Kaiserslautern followed by a Postdoc time at the Chalmers University of Technology in Göteborg, Sweden. She is interested in verification and test methods for variant-rich and evolving software systems.