Preference Models and Their Elicitation and Analysis for Context-Aware Applications

Radosław Klimek

Abstract. The work concerns building preference models when gathering system requirements and their formal analysis using the deductive approach. The selected UML diagrams are used for elicitation of preference models and temporal logic for their specification and verification. The proposed method makes preference modeling more reliable in the phase of requirements engineering. Preference models are based on predefined patterns. It enables the process of generating logical specifications for preference models to be automated.

Keywords: context-aware systems, pervasive applications, requirements engineering, preference models, preference patterns, temporal logic, deductive reasoning.

1 Introduction

Preference modeling and prediction for context-aware applications are crucial in software engineering. The construction of preference models is particularly important in systems related to pervasive and ubiquitous computing. Preference modeling constitutes a kind of bridge between a support-oriented user and a system which is able to provide the support. Thus, context-aware systems must adapt their behavior in response to the user's needs. The requirements engineering phase seems especially convenient for elicitation of preference models. Another motivation for the work is the lack of tools for automatic extraction of logical specifications based on temporal logic for preference models. The main contribution is automation of the generation process for logical specifications of preference models. Preference models are obtained in the process of requirements engineering aimed at, among others,

Radosław Klimek
AGH University of Science and Technology,
al. A. Mickiewicza 30, 30-059 Krakow, Poland
e-mail: rklimek@agh.edu.pl

A. Gruca et al. (eds.), *Man-Machine Interactions 3*,
Advances in Intelligent Systems and Computing 242,
DOI: 10.1007/978-3-319-02309-0_38,

building preference models through a proposed quasi methodology, which includes some UML diagrams.

Preference modeling needs formalization and it is discussed in some works, e.g. fundamental and cited work by Öztürk et al. [7]. The model of preferences may be constructed using fuzzy sets, classical logic and many-valued logics. Classical logic, and particularly rule-based systems, are especially popular, c.f. work by Fong et al. [4]. Non-classical logics, and especially temporal logic, are less popular. On the other hand, temporal logic is a well-established formalism for describing the reactiveness of the system. Typical pervasive applications should be characterized by reactivity and flexibility in adapting to changes on the user side. These changes may result from recognized and predefined preferences. Therefore, it seems that this reason is important enough to include temporal logic in the considerations for preference models. The temporal approach seems underestimated in preference modeling. Work by van Eijck [2] discusses modal logics for preference and belief. It concerns theoretical issues. A (very) preliminary version of this work is work [5]. In work [6], state-space reduction using preference models for the agent domain is discussed. Requirements engineering issues for preference models, as they are presented in the work, are discussed for the first time.

2 UML Diagrams

Some aspects of UML diagrams, e.g. [8,9], are discussed below. They can be applied for building a requirements model, however they may also be used for constructing preference models. The presented point of view covers some well-defined steps and may constitute a kind of quasi methodology and is shown in Fig. 1. Step by step, more and more information about the preference model is obtained and in the end of the procedure, it is also logically analyzed. The *use case diagram* consists of *actors* which create the system's environment and *use cases* which are services and functionalities used by actors. The diagram is a rather descriptive technique compared with the other UML diagrams. Each use case has its *scenario* which is a brief narrative that describes an expected use of the system. The scenario describes a basic and possible alternative flows of events. It may describe activities or preferences. An *activity* is a computation with its internal structure. From the point of view of the approach presented here, scenarios may also be used for preference discovery. The *activity diagram* enables modeling of activities. It is a graphical representation of the workflow, showing the flow of control from one activity to another.

More formally, use case diagrams UCD contain many use cases UC which describe the desired functionality of a system and expected preferences, i.e. $UC_1, ..., UC_i, ..., UC_l$, where $l > 0$ is the total number of use cases created during the modeling phase. Each use case UC_i has its own scenario, which identifies its activities and preferences. Thus, every scenario contains some activities $a_1, ..., a_k, ..., a_m$, and some preferences $p_1, ..., p_k, ..., p_n$ where $m > 0$ and $n > 0$. The level of formalization presented here, i.e. when discussing use cases and their scenarios, is intentionally not very high. This assumption seems realistic since this is an initial phase

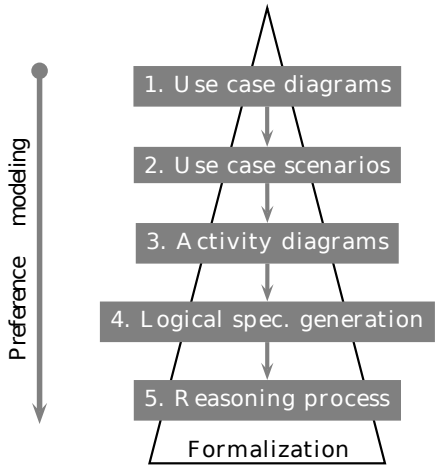

Fig. 1 Preference modeling and analysis

of system development and one of the most important things in this approach is to identify the activities and preferences when creating scenarios, since these objects will be used when modeling activity diagrams.

3 Preference Models

Temporal Logic. TL is a valuable formalism, e.g. [3, 10], which has strong application for specification and verification of models. It exists in many varieties, however, considerations in this paper are limited to the *Linear Temporal Logic* LTL, i.e. logic, for which the time structure is considered linear. Considerations in this paper are also limited to the *smallest temporal logic*, e.g. [1], also known as temporal logic of the class K.

The preference model proposed in the work is based on predefined patterns. A *pattern* is a predefined solution for a special context, which are preference issues. Preferences form logical rules expressed in temporal logic. Patterns constitute a kind of primitives and are generally indicated as $pat()$ (parameters are included in parentheses), where *pat* is a name of a given pattern. The following patterns are considered: *Branch*, *SimpleBranch* and *Sequence*, c.f. [5, 6]. The process of the preference elicitation is the following:

1. use cases and use case diagrams are modeled first; they provide a high level and functional-oriented description of the system;
2. use case scenarios, carefully written in a text form, are instances of use cases representing the expected uses of a system; preferences are identified here for the first time, i.e. $p_1, ..., p_n$;

3. activity diagrams enable modeling previously identified preferences using predefined patterns.

Patterns of behaviors and preferences can be nested. It follows from the situation of multi-stage decision-making. A *basic set of patterns* Σ is a set of predefined preference patterns, for example a set of three patterns, i.e. $\Sigma = \{Branch, SimpleBranch, Sequence\}$, is considered here. Let us define *temporal properties* $\Pi(\Sigma)$ over predefined patterns Σ. An *elementary set* of formulas over atomic formulas $a_{i,i=1,...,n}$ is denoted $pat(a_i)$, or simply $pat()$, as a set of temporal logic formulas $\{f_1, ..., f_m\}$ such that all formulas are syntactically correct. The set $Branch(f_1, f_2, f_3) = \{c(f_1) \Rightarrow \Diamond f_2 \wedge \neg\Diamond f_3, \neg c(f_1) \Rightarrow \neg\Diamond f_2 \wedge \Diamond f_3, \Box\neg(f_1 \wedge (f_2 \vee f_3))\}$ describes the properties of the Branch pattern and $SimpleBranch(f_1, f_2) = \{c(f_1) \Rightarrow \Diamond f_2, \neg c(f_1) \Rightarrow \neg\Diamond f_2, \Box\neg(f_1 \wedge f_2)\}$ – of the SimpleBranch pattern. The set $Sequence(f_1, f_2) = \{f_1 \Rightarrow \Diamond f_2, \Box\neg(f_1 \wedge f_2)\}$ defines the Sequence pattern. Formulas f_1, f_2 etc. are atomic formulas for a pattern. They constitute a kind of formal arguments for a pattern. $c(f)$ means that the logical condition associated with activity f has been evaluated and is satisfied.

The entire preference model can be written in the form of the *logical expression* W_L, which is similar to the well-known regular expressions. The goal is to write preferences in a literal notation which allows to represent complex and nested structures, c.f. $Sequence(Branch(a,b,c), SimpleBranch(d,e))$. A sequence of two branchings is considered, i.e. an ordinary and a simple one. a and d are conditions in this expression. Individual preferences may belong to a set of preferences R, i.e. $R = \{r_1, r_2, ...r_n\}$, where every r_i is a preference expressed as a single expression. A set of temporal logic formulas can be generated (extracted) from (complex) logical expressions. These formulas describe preference properties and their desired behavior. Let us summarize the approach. For every use case UC and its scenario, which belongs to any use case diagram UCD, some activity diagrams AD are developed. Preferences $r_1, ...r_n$ obtained from activity diagrams are modeled using atomic preferences $p_1, ...p_m$ which are identified when building a use case scenario. Preferences are composed only using the predefined patterns.

Building a logical model for the gathered preferences in the form of temporal logic formulas enables examination of both semantic contradiction and correctness of the logical model, according to some properties. The architecture of a system for automatic inference on preference models is proposed in works [5,6]. The inputs of the system are logical expressions R and a predefined set of basic preference patterns Σ together with their temporal properties $\Pi(\Sigma)$ which are predefined and fixed. The output is a logical specification L understood as a set of temporal logic formulas. The outline of the generation algorithm is as follows:

1. at the beginning, the logical specification is empty, i.e. $L = \emptyset$;
2. the most nested pattern or patterns of every R are processed first; then, less nested patterns are processed one by one, i.e. patterns that are located more towards the outside;

3. if the currently analyzed pattern consists only of atomic formulas, the logical specification is extended, by summing sets, by formulas linked to the type of the analyzed pattern $pat()$, i.e. $L = L \cup pat()$;
4. if any argument is a pattern itself, then the logical disjunction of all its arguments, including nested arguments, is substituted in place of the pattern.

The above algorithm refers to similar ideas in work [6]. Examples for the algorithm are presented in the next section.

The standard taxonomy of system properties includes liveness and safety, which are adapted here to the domain of preferences:

- *liveness* means that some preferences might be achieved in the whole preference model, e.g. $\Diamond p_3$ or $p_1 \Rightarrow \Diamond p_5$;
- *safety* means that some preferences, perhaps a logical combination of a subset, are avoided, e.g. $\Box\neg(p_4 \wedge \neg p_9)$.

where $\{p_1, ..., p_9\}$ are atomic preferences belonging to the entire preference world identified when preparing use case scenarios.

4 Illustration of the Approach

Let us consider a simple yet illustrative example. The example concerns a bicycle shop offering internet sales. A typical and sample use case diagram is shown in Fig. 2. It consists of two actors and three use cases, UC_1, UC_2 and UC_3, modeling the system for a bike shop.

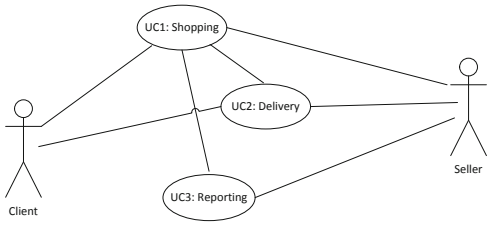

Fig. 2 A sample use case diagram "BicycleShopping"

Every use case has its own scenario, c.f. Fig. 3. They contain atomic preferences which are identified when preparing the scenario. In the case of the UC_1 scenario the preference refers only to the color of a bike and to its derailleurs. The scenario identifies atomic preferences: "YellowColor" (or "a" as alias), "SelectYellow" (b), "SelectOther" (c), "DoubleDerailleur" (d), "FrontRearDerailleur" (e) and "RearDerailleur" (f). In the case of the UC_2 scenario, the preference refers to the form of payment and to the delivery method. Card payment and a self pick-up are preferred. When the bikes are shipped, then a courier company is preferred.

UC1: Shopping

Scenario:

1. **Prepare catalogue**
2. **"YellowColor"(?) of bike is preferred, i.e. "SelectYellow" else "SelectOther" color**
3. **"DoubleDerailleur"(?) for a bike is preferred then select bike with "FrontRearDerailleur" else at least "RearDerailleur"**
4. **Calculate the price**
5. **Update a store data base**

UC2: Delivery

Scenario:

1. **Prepare "Receipt" for a client**
2. **Complete payment depending on "CardPayment"(?), i.e. "PayCard" (preferred) or "PayBankTransfer"**
3. **Prepare bike for a client**
4. **"SelfPickUp"(?) of a bike is preferred then get "Acknowledgement"**
5. **If not self pick-up then in the case of "FastShipping"(?) choose "Courier", which is preferred, else prepare ordinary "ParcelPost"**
6. **Update a store data base**

Fig. 3 A scenario for the use case UC_1 "Shopping" (left) and a scenario for the use case UC_2 "Delivery" (right)

The scenario contains nested preferences. It also identifies atomic preferences: "Receipt" (or "g" as alias), "CardPayment" (h), "PayCard" (i), "PayBankTransfer" (j), "SelfPickUp" (k), "Acknowledgement" (l), "FastShipping" (m), "Courier" (n) and "ParcelPost" (p). One of the most important things in this phase is to identify preferences when creating scenarios and have general and informal idea about them. Dynamic aspects of preferences are to be modeled strictly when developing activity diagrams.

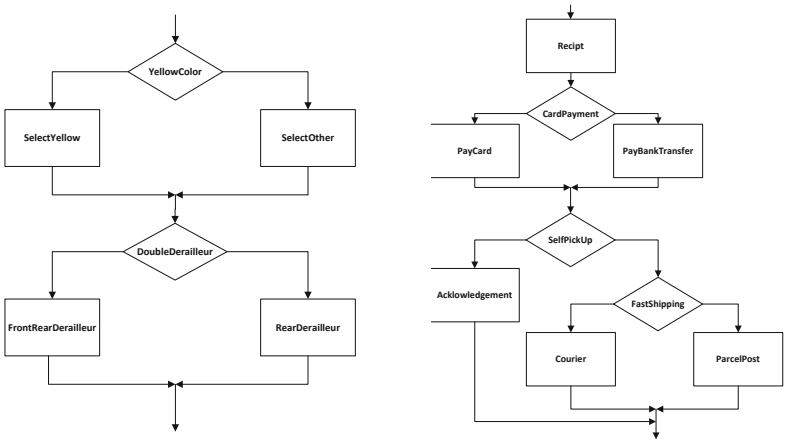

Fig. 4 An activity diagram AD_1 for the "Shopping" scenario (left) and an activity diagram AD_2 for the "Delivery" scenario (right)

For every use case scenario, an activity diagram is created. The activity diagram workflow is modeled only using atomic preferences which are identified when building a use case scenario. Furthermore, workflows are composed only using the predefined design patterns shown in Fig. 4. Nesting of patterns is acceptable. A sample activity diagram AD_1 is shown in Fig. 4. It models preferences for the UC_1 use case shown in Fig. 2 and activities from the scenario in Fig. 3. It contains a sequence of two branches. The activity diagram for the UC_2 scenario is a bit more complex, and is also shown in Fig. 4. This step completes the phase of modeling preferences.

The next phase performs automatic generation of logical expressions from activity diagrams. The logical expression for AD_1 is as follows

$$r_1 = Seq(Branch(YellowColor, SelectYellow, SelectOther), \\ Branch(DoubleDerailleur, FrontRearDerailleur, RearDerailleur)) \quad (1)$$

The logical expression for activity diagram AD_2 is

$$r_2 = Seq(Receipt, Seq(Branch(CardPayment, PayCard, \\ PayBankTransfer), Branch(SeflPickUp, Acknowledgment, \\ Branch(FastShipping, Courier, ParcelPost)))) \quad (2)$$

Further steps should include generation of logical specifications for preference models. These are generated from logical expressions 1 and 2 using the algorithm presented in the Section 3. (Replacing propositions by Latin letters is a technical matter and it follows from the limited size of the work.) For example, for the logical expression r_1, the generation process progresses as follows. For first branch it gives $L = \{c(a) \Rightarrow \Diamond b \wedge \neg\Diamond c, \neg c(a) \Rightarrow \neg\Diamond b \wedge \Diamond c, \Box\neg(a \wedge (b \vee c))\}$. Considering the second branch gives $L := L \cup \{c(d) \Rightarrow \Diamond e \wedge \neg\Diamond f, \neg c(d) \Rightarrow \neg\Diamond e \wedge \Diamond f, \Box\neg(d \wedge (e \vee f))\}$ The sequence of these branches gives $L := L \cup \{(a \vee b \vee c) \Rightarrow \Diamond(d \vee e \vee f), \Box\neg((a \vee b \vee c) \wedge (d \vee e \vee f))\}$. Thus, the final logical specification for r_1 is $L = \{c(a) \Rightarrow \Diamond b \wedge \neg\Diamond c, \neg c(a) \Rightarrow \neg\Diamond b \wedge \Diamond c, ...\Box\neg((a \vee b \vee c) \wedge (d \vee e \vee f))\}$ The logical specification for logical expression r_2 can be generated in a similar way. When the whole logical specification is generated, it can by analyzed both for contradiction and for some properties (liveness, safety), c.f. [5, 6].

5 Conclusions

The work presents a new approach to obtaining preference models when gathering the requirements of the system. Preference models are transformed to logical specifications expressed in temporal logic. That enables formal analysis and verification of preferences. Further works may include the implementation of the modeler and the logical specification generation module. It is a key issue for context-aware systems, which must adapt their behavior in response to the user's needs.

Acknowledgements. This work has been partially financed by the European Union, Human Capital Operational Programme, SPIN project no. 502.120.2066/C96.

References

1. Chellas, B.F.: Modal Logic: An Introduction. Cambridge University Press (1980)
2. van Eijck, J.: Yet more modal logics of preference change and belief revision. In: Apt, K.R., van Rooij, R. (eds.) New Perspectives on Games and Interaction, pp. 81–104. Amsterdam University Press (2008)
3. Emerson, E.A.: Temporal and modal logic. In: van Leeuwen, J. (ed.) Handbook of Theoretical Computer Science, vol. B, pp. 995–1072. Elsevier, MIT Press (1990)
4. Fong, J., Indulska, J., Robinson, R.: A preference modelling approach to support intelligibility in pervasive applications. In: Proceedings of 9th IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops) - 8th IEEE Workshop on Context Modeling and Reasoning (CoMoRea 2011), pp. 409–414. IEEE (2011)
5. Klimek, R.: Temporal preference models and their deduction-based analysis for pervasive applications. In: Benavente-Peces, C., Filipe, J. (eds.) Proceedings of 3rd International Conference on Pervasive and Embedded Computing and Communication Systems (PECCS 2013), pp. 131–134. SciTe Press (2013)
6. Klimek, R., Wojnicki, I., Ernst, S.: State-space reduction through preference modeling. In: Rutkowski, L., Korytkowski, M., Scherer, R., Tadeusiewicz, R., Zadeh, L.A., Zurada, J.M. (eds.) ICAISC 2013, Part II. LNCS (LNAI), vol. 7895, pp. 363–374. Springer, Heidelberg (2013)
7. Öztürké, M., Tsoukiàs, A., Vincke, P.: Preference modelling. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) Multiple Criteria Decision Analysis: State of the Art Surveys, vol. 78, pp. 27–59. Springer (2005)
8. Pender, T.: UML Bible. John Wiley & Sons (2003)
9. Rumbaugh, J., Jacobson, I., Booch, G.: The Unified Modeling Language Reference Manual, 1st edn. Addison-Wesley (1999)
10. Wolter, F., Wooldridge, M.: Temporal and dynamic logic. Journal of Indian Council of Philosophical Research 27(1), 249–276 (2011)