A NFR-based Framework for User-Centered Adaptation

Fabiano Dalpiaz[1], Estefanía Serral[2], Pedro Valderas[2], Paolo Giorgini[1], and Vicente Pelechano[2]

[1] DISI, University of Trento
{dalpiaz,pgiorgio}@disi.unitn.it
[2] PROS, Universitat Politècnica de València
{eserral,pvalderas,pele}@dsic.upv.es

Abstract. Pervasive environments support users' daily routines in an invisible and unobtrusive way. To do so, they include a technical pervasive infrastructure, which is aware of and adaptive to both the operational context and the users at hand. Non-Functional Requirements (NFRs) have been effectively used to inform decision-making in software engineering: functional alternatives are compared in terms of their contribution to NFRs satisfaction. In this work, we consider user preferences over NFRs as a key driver for the adaptation of a pervasive infrastructure. We devise a model-driven framework for building pervasive systems that maximize fitness with the context and the user. Our contributions are: (i) *adaptive task models*, a conceptual model to describe user routines that accounts for user preferences over NFRs; and (ii) an adaptation framework, which uses our models at runtime to guide a pervasive infrastructure in adapting its behaviour to user preferences and context.

Keywords: pervasive environments; self-adaptation; NFRs

1 Introduction

In pervasive computing [17], technical systems are deployed in the environment—the so-called *pervasive environment*—so as to support humans in their daily activities. Crucially, pervasive environments have to remain, while executing, invisible and unobtrusive to users. The technical infrastructure of pervasive environments (*pervasive infrastructure*) effects changes in the environment and suggests appropriate activities to the users. While being guided by this system, the user should not realize that the system is "thinking" on her behalf.

Task models [14] are a modelling language to represent user routines [18] (sets of habitually performed tasks). They are an example of executable conceptual models, as they hierarchically specify and temporally relate the tasks a system should execute for supporting a user in the conduction of her daily routines. These models have been successfully adopted in model-driven pervasive infrastructures [18].

However, task models provide limited adaptation to user preferences about non-functional properties. In order to be unobtrusive and invisible, the system has to execute routines that support courses of action the user finds natural to her (i.e., that match her preferences). If the user is interested in carbon emissions reduction, the system should minimize heating usage and suggest going to work on foot. However, if she has scheduled early meetings and is late, the system should recommend fast transportation means such as driving.

We investigate how user preferences over non-functional properties can be taken into account by a pervasive infrastructure. Our approach relies upon Non-Functional Requirements (NFRs) [13]. NFRs have been successfully used to inform decision-making by choosing the alternative that maximizes the satisfaction of qualities. Based on successful applications in requirements models [23], architectures [5], and business processes [15], we investigate their effective usage NFRs with task models.

In this paper, we extend task models and propose a model-driven framework that enables a pervasive infrastructure to adapt its behaviour to the user preferences and the current context. Our contributions are as follows:

– *adaptive task models*, a modelling language that enriches task models with NFRs. The model is created at design-time, and used by the system at runtime to decide upon how to adapt its behaviour. User preferences over NFRs are captured by our proposed *contextual preference model*: each user specifies, in a context-dependent way, the priority she assigns to each NFR;
– *an adaptation framework* for building pervasive infrastructures that exploit adaptive task models and contextual preference models at runtime.

The paper is organised as follows. Sec. 2 presents our baseline. Sec. 3 introduces adaptive task models and the contextual preference model. Sec. 4 proposes an adaptation framework that exploits our proposed models at runtime. Sec. 5 discusses related work, draws our conclusions and outlines future directions.

2 Research Baseline: Task Models

We build on top of the task models by Serral et al. [18], which specify how a pervasive infrastructure can support its users in carrying out everyday activities. Task models are inspired by Hierarchical Task Analysis (HTA) [20], which constructs a task tree that refines a high-level task into a set of executable ones.

Running example. A smart-home pervasive environment supports the daily routines of the home inhabitants through a set of pervasive services [18] that are interfaced with sensors and effectors. Consider the following routine: “Every working day, the system turns on the bathroom heating at 7:50 a.m. to make it warm enough for Bob to take a shower. At 8:00 a.m., the system makes a wake-up call, repeating it until Bob wakes up. Then, the room is illuminated and Bob is notified about the weather. Afterwards, when Bob enters the kitchen, the system makes a coffee, and suggests him the best way to go to work.” □

Fig. 1 illustrates the “Waking Up” routine. The root task is broken down into simpler tasks by means of two task refinement constructs: *exclusive refinement*

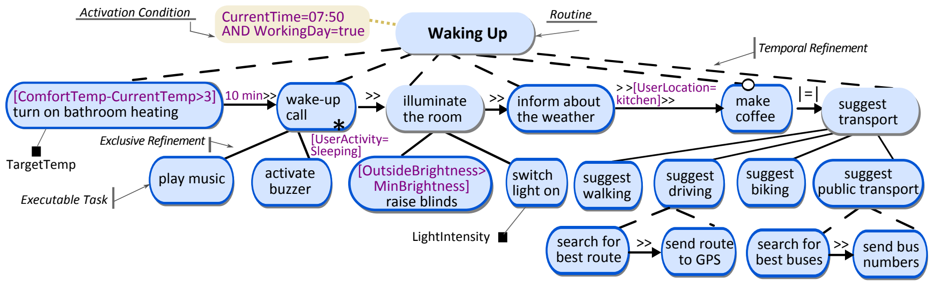

Fig. 1. Task model representing the "Waking Up" user routine

and *temporal refinement*. Exclusive refinement (graphically, a solid line) decomposes a task into a set of subtasks so that exactly one subtask will be executed. Temporal refinement (graphically, a dashed line) also decomposes a task into subtasks; however, all the subtasks shall be performed following a specific order which is depicted by the arrows between sibling tasks. Temporal constraints employ Concurrent Task Trees (CTT) operators [14]. For example, in Fig. 1:

– *Enablement* ($T_1 \gg T_2$): task T_2 is triggered when task T_1 finishes. For instance, the system has to illuminate the room after waking up the user;
– *Task Independence* ($T_1 \,|{=}|\, T_2$): T_1 and T_2 can be performed in any order. For instance, "make coffee" and "suggest transport" are temporally independent.

Task refinement ends when every leaf task is linked to a pervasive service (controlled by the pervasive infrastructure), which will execute the task. For example, "raise blinds" is executed by a pervasive service controlling the blinds engine.

A routine can be carried out through alternative sets of tasks depending on the current state of the context (the "situation" [11]). Situations are used to indicate the relationship between context and routine execution (see Fig. 1):

– *Activation condition*: it is associated with the root task of each routine. It indicates the situation in which the routine is activated. For instance, the "Waking Up" routine is to be executed every working day at 7:50 a.m.;
– *Task precondition*: it can be associated with a task to indicate that its execution depends on whether a situation (between square brackets) holds. For instance, the bathroom heating shall be turned on only if the difference between comfort and current temperature is greater than three degrees.
– *Iterative task*: it is executed repeatedly while the situation associated with the task holds. These tasks are graphically marked with an asterisk. For instance, task "wake-up call" is iterated while the user sleeps, and the iteration stops as soon as the user wakes up;
– *Temporal constraints:* the following relationships indicate that the execution of two tasks (linked by an arrow) is subject to a temporal constraint:
  • $T_1 \gg [s] \gg T_2$: after the completion of T_1, T_2 is started as soon as situation s holds. In Fig. 1, the system makes coffee after informing about

the weather, as soon as the user is in the kitchen (situation "UserLocation=kitchen" holds). Note that s could already hold when T_1 ends;
- $T_1\ t \gg T_2$: after the completion of T_1, T_2 is started as soon as the time period t has elapsed. For instance, 10 minutes after turning the bathroom heating on, the system shall execute task "wake-up call".

3 Adaptive Task Models

We propose an extended version of task models (*adaptive task models*) to describe system behaviour that takes into account both the *personal context* [22], i.e., the individual requirements and preferences of specific users, and the *physical and social context*, i.e., observable characteristics of the environment that the system can monitor (e.g., who is in the room, temperature, closed and open doors).

Our extension enables not only adaptation to the preferences of different users—while one may be more concerned with energy efficiency, another may give priority to user comfort—, but also to the changing preferences of a specific user. For instance, if a user is in a hurry, she may favour efficiency over comfort; when not at home, instead, she may be more interested in energy saving.

Fig. 2 depicts the meta-model of adaptive task models. The red-coloured classes show our proposed contextual preference model (Sec. 3.1), which indicates user preferences over NFRs. The white-coloured classes represent the extended task model itself: we introduce optional and parametric tasks (Sec. 3.2), as well as task contributions to NFRs (Sec. 3.3). The green-coloured classes represent the context model (from previous work [18]).

3.1 Contextual Preference Model for NFRs

Each user has different preferences, which depend on the situation and vary over time. To represent users preferences over NFRs, we propose the contextual preference model (which extends [11]). Our model enables analysts to define the relevant NFRs and the priority assigned by each user in different contexts.

Table 1. Partial contextual preference model for user Bob

User Comfort (UC) :	⟨UserLocation≠Home, 0⟩
	⟨UserLocation=Home ∧ UrgentTasks=false, 0.7⟩
	⟨UserLocation=Home ∧ UrgentTasks=true, 0.5⟩
User Efficiency (UE) :	⟨UserLocation≠Home, 0⟩
	⟨UserLocation=Home ∧ UrgentTasks=true, 1⟩
	⟨UserLocation=Home ∧ UrgentTasks=false, 0.3⟩
Energy Efficiency (EE) :	⟨UserLocation≠Home, 0.9⟩
	⟨UserLocation=Home, 0.4⟩

For each NFR, a set of couples consisting of a situation s and a weight w (a real number in the range [0,1]) is specified. Each couple indicates that,

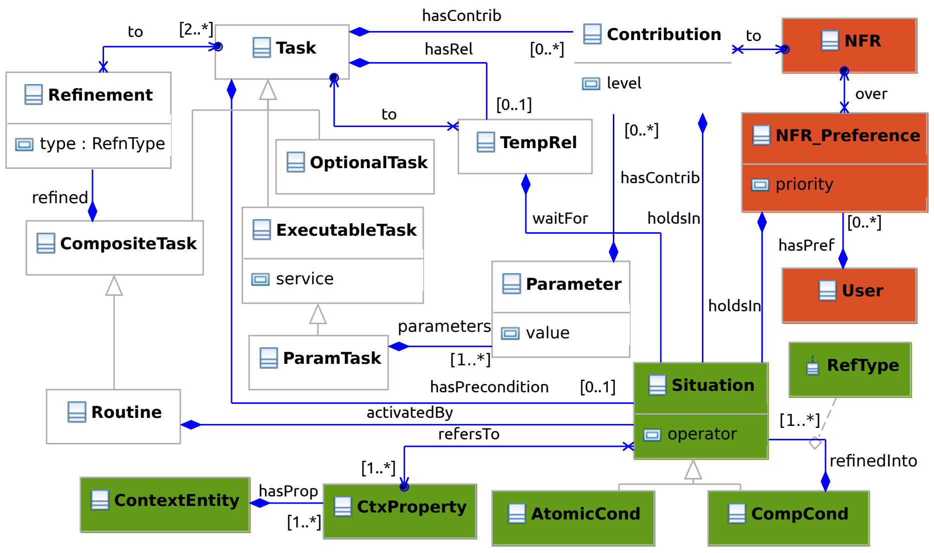

Fig. 2. Overview of adaptive task models

when situation s holds, the NFR has priority w. Value 0 indicates minimum importance, while value 1 indicates maximum importance. The situations for a given NFR have to be mutually exclusive. Table 1 shows part of the contextual preference model for Bob. The weight of NFR user comfort is 0 when he is not at home; if Bob is at home, it is 0.7 if he has no urgent tasks, 0.5 otherwise.

3.2 Task Models with Optional and Parametric Tasks

Our approach to adaptation is model-driven in general, and NFR-driven in particular. The system adapts by choosing an adequate course of action from its task model, on the basis of contextual factors and user preferences over NFRs. To such extent, we enrich task models with optional and parametric tasks.

Optional tasks are not essential to accomplish a specific routine. Their execution depends on user preferences over NFRs. For instance, task "make coffee" is optional. If the user has a meeting early in the morning, the NFR user efficiency will have a high priority, and task "make coffee" will not be executed. Note that optional is different from contextual: a contextual task is executed only if the context precondition holds, while an optional task relies on preferences. Graphically (see Fig. 3), optional tasks are represented by drawing a hollow circle to the incoming refinement link (like optional features in feature models [10]).

Parametric tasks are leaf tasks in the task model whose execution can be tuned by adjusting the values of their parameters. This tuning is intended to maximise satisfaction of user preferences. For instance, task "turn on bathroom heating" can be tuned by adjusting the target temperature, while "switch bedroom light on" by setting light intensity. Depending on the value of their parameters, the tasks will have different impacts on the NFRs. Graphically (see Fig. 3), parameters are labels—representing the parameter name—connected to leaf tasks by a line ending with a square-shaped arrow.

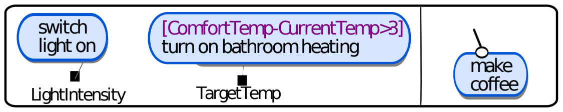

Fig. 3. Graphical representation of parametric and optional tasks

3.3 Linking Tasks to NFRs via Contributions

Variation points enable adaptability in a model-driven system: they are the loci in the model wherein alternative decisions are taken by the system (depending on the current context). In this section we explain how task contributions to NFRs can be exploited in adaptive task models to drive system adaptation so as to choose the alternative that fits best with user preferences.

Like in goal analysis [9], analysts indicate the contribution of each task to individual NFRs in the range [-1,+1]. Let c be such contribution. A task can be neutral (c=0), provide a negative contribution (c<0), or a positive contribution (c>0). If no value is specified, we consider a neutral contribution relation.

While expressing contributions, analysts have to distinguish between tasks where the system automates an activity (automation) and tasks in which the system suggests the user a specific course of actions (recommendation):

- *automation*: the contribution quantifies the direct impact of task execution by the system. For example, contributions for task "raise blinds" refer to the system action of turning on the blinds engine;
- *recommendation*: the contribution evaluates the indirect impact of having the user following the suggestion. For example, contributions for task "suggest walking" refer to the impact of accepting the suggestion and walking to work, and not to the action of recommending the user.

We require contributions to be specified in correspondence of all variation points in a routine. We suppose that the running system explores the task model for a routine in a top-down fashion, and takes decisions about which alternative to choose whenever it encounters a variation point.

Adaptive task models include three variation point types: (i) exclusive refinement: one subtask is to be selected and executed; (ii) optional tasks: can be either executed or skipped; and (iii) parametric tasks: parameters can be tuned, leading to different runtime behaviours. We detail each variation point type:

Exclusive refinement: the system has to choose the best alternative subtask by comparing their contribution to NFRs. For a non-executable task, the contribution approximates the level of contribution of the abstract task, irrespective of the specific executable tasks that refine it. Contributions can be context-dependent. For instance, consider NFR user comfort and task "suggest driving" in the "Waking Up" routine. The contribution of this task could be +0.5 if the user lives and works in the city outskirts, -0.5 if the user either works or lives in the city centre, where traffic jams are very likely to occur.

Optional task: the decision is whether to execute or skip the task, depending on the contribution to NFRs and user preferences. Contribution to NFRs is expressed as explained for the exclusive refinement variation point. The rule of thumb is that the task is carried out if the weighted contribution to NFRs is positive (>0), and is skipped otherwise. Take, for instance, task "make coffee", and suppose its contribution to NFR user comfort is +0.6 if the user has no early meetings (situation "UrgentTasks" does not hold), -0.8 otherwise; and its contribution to NFR User Efficiency is -0.4. Take Bob's preferences from Table 1. Depending on the current context, the task is executed or skipped:

– "UrgentTasks=true": user comfort has weight 0.5, user efficiency 1.0. The average contribution value is $\frac{(-0.8*0.5)+(-0.4*1)}{0.5+1} = -0.53$; being negative, the task is skipped;
– "UrgentTasks=false": user comfort has weight 0.7, user efficiency 0.3. The average contribution value is $\frac{(0.6*0.7)+(-0.4*0.3)}{0.7+0.3} = +0.3$; being positive, the task will be executed by the system.

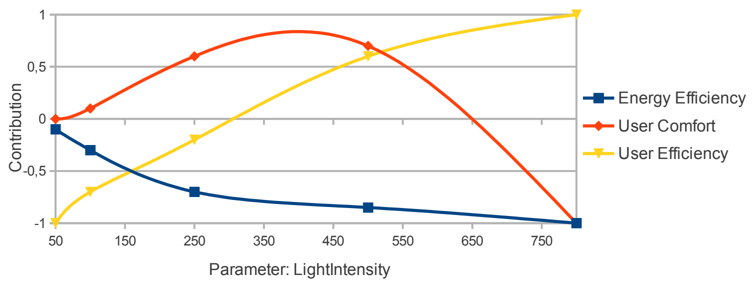

Fig. 4. Interpolation functions for NFR contributions of the task "switch light on"

Parametric task: the system has to tune the parameters for optimizing NFRs. If the task depends on multiple parameters, so as to simplify the specification of contributions, we suppose the analysts will merge these parameters into a single numeric parameter in a discrete interval. Contribution values are assigned for a set of known values, obtained either by expertise, through interviews, from data sheets, or via measurements. The system will determine the contribution for the missing values using interpolation functions [19] (e.g., polynomial, spline, cubic).

Take, for instance, task "switch light on". Depending on the light intensity, NFRs energy efficiency, user comfort, and user efficiency receive different contributions. In Fig. 4, the analysts have specified contributions to the three NFRs for different light intensity values (50, 100, 250, 400, 800 lux), based on her own experience and the light bulb data sheet. A spline interpolation has been applied to compute the contribution values for the missing light intensity values.

When a task is both parametric and a subtask in an exclusive refinement, parametric task contributions are considered instead of contextual contributions.

4 Executing Adaptive Task Models

Adaptive task models are machine-processable and are executable models. At runtime, they drive the adaptive behaviour of a pervasive infrastructure. In this way, all the efforts invested at design time are reused at runtime providing new opportunities for adaptation capabilities without increasing development costs.

The adaptation process is activated by triggering events, which define *when* the system should adapt. These triggering events are: changes in user preferences, task execution faults, plan failures or context evolution.

The pervasive infrastructure will use the information about the occurrence of a trigger in the next execution (instance) of a routine. In such next instance, the plan that supports the user best—i.e., maximizes NFRs—will be selected. A plan consists of a set of executable tasks in the routine and ordering constraints between those tasks that, together, carry out the routine.

For instance, consider it's a hot Monday of September, and Bob has urgent tasks at work. In this context, NFRs weights are: user comfort = 0.5, user efficiency = 1, energy efficiency = 0.4. The pervasive infrastructure executes the adaptive task model of Fig. 1 as follows: the root task is temporally refined, so its subtasks are examined. The precondition of task "turn on the heating" does not hold, thus the task is skipped. After ten minutes, a wake up call is made. The task is exclusively refined. The buzzer option is chosen by comparing the weighted contributions to NFRs. Since user efficiency has priority over comfort, task "activate buzzer" is executed. Bob awakes immediately. The room is illuminated by raising the blinds, since the contribution of this task to user efficiency is higher than the contribution of "switch light on", no matter its tuning. Bob is informed about the weather by executing such task. When Bob enters in the kitchen, driving is suggested, as this option has the best contribution to user efficiency. Note that the optional task "make coffee" is not executed because Bob has urgent tasks and is in a hurry (see Sec. 3.3).

5 Discussion

We have proposed adaptive task models, an executable modelling language that a pervasive infrastructure can use to support users in their daily routines. These models do not only support alternatives to carry out a routine, but also include the decision-making rationale. Moreover, by handling preferences over NFRs as a separated model, a specific routine can be presented to the user in different ways just by changing the contextual preference model, without altering the routine.

We have also devised a user-centric adaptation framework (more details in [7]) that uses adaptive task models at runtime to adapt system behaviour. While automating user routines, the system adapts its behaviour by choosing a course of action that maximizes user preferences over NFRs.

The use of NFRs to drive decision-making has been widely explored in Goal-Oriented Requirements Engineering (GORE). Most GORE approaches rely upon (variants of) the NFR framework [13] or the $i*$ framework [23], and exploit the concept of soft-goal to represent NFRs and reason about NFR satisfaction.

Chung et al. [5] use a NFR graph to select among alternative architectural designs. Adaptive software systems (e.g., [12]) use soft-goals to choose the configuration that maximizes the satisfaction of a set of NFRs. We also rely on optimizing NFRs; however, unlike those approaches, we account for the priorities of each user, and we allow for contextual contributions.

Brown et al. [3] exploit a goal-oriented specification to define adaptation requirements, i.e. how the system switches from one configuration to another. In a similar spirit, Souza et al. [21] define awareness requirements as meta-requirements to drive adaptations. Our framework embodies an adaptation requirement, i.e. the optimization of user preferences over NFRs.

Some approaches [16, 8] explore contextual variations in business process models. Interestingly, in [8], context analysis [2] is used to specify context. Such approach could be exploited to define situations. Another interesting direction is assessing the suitability of BPM languages for representing user routines.

Other approaches use feature models [4, 1] to describe architectural configurations, each consisting of components that are activated depending on the current context. In addition to contextual factors, we consider user preferences over NFRs so as to adapt the system. Also, we do not focus on individual components, but on the adaptation of complex system behaviours (user routines).

Future work includes the development of a pervasive software infrastructure based on our adaptation framework. We intend to rely upon previous work on pervasive infrastructures [18] and goal-oriented adaptation [6]. The effectiveness of the infrastructure in adapting to users will be assessed through case studies.

Acknowledgement

This work has been developed with the support of: the European Union Seventh Framework Programme (FP7/2007-2013) under grants no 257930 (Aniketos) and 256980 (Nessos); and MICINN under the project EVERYWARE TIN2010-18011.

References

1. M. Acher, P. Collet, F. Fleurey, P. Lahire, S. Moisan, and J. P. Rigault. Modeling Context and Dynamic Adaptations with Feature Models. In *Proc. of Models@run.time 2009*. LNCS, 2009.
2. R. Ali, F. Dalpiaz, and P. Giorgini. A Goal-based Framework for Contextual Requirements Modeling and Analysis. *Requirements Engineering*, 15(4):439–458, 2010.

3. G. Brown, B. H. C. Cheng, H. Goldsby, and J. Zhang. Goal-oriented Specification of Adaptation Requirements Engineering in Adaptive Systems. In *Proc. of SEAMS '06*, pages 23–29. ACM, 2006.
4. C. Cetina, P. Giner, J. Fons, and V. Pelechano. Autonomic Computing through Reuse of Variability Models at Runtime: The Case of Smart Homes. *IEEE Computer*, 42:37–43, 2009.
5. L. Chung, B. Nixon, and E. Yu. Using Non-Functional Requirements to Systematically Select among Alternatives in Architectural Design. In *Proc. of IWASS'95*, page 3143. ACM, 1995.
6. F. Dalpiaz, P. Giorgini, and J. Mylopoulos. Adaptive Socio-Technical Systems: a Requirements-driven Approach. *Requirements Engineering*, 2012. To appear.
7. F. Dalpiaz, E. Serral, P. Valderas, P. Giorgini, and V. Pelechano. A NFR-based Framework for User-Centered Adaptation. TR DISI-12-022, DISI, University of Trento, 2012.
8. J. L. de la Vara, R. Ali, F. Dalpiaz, J. Sanchez, and P. Giorgini. COMPRO: A Methodological Approach for Business Process Contextualisation. In *Proc. of CoopIS 2010*, pages 132–149, 2010.
9. P. Giorgini, J. Mylopoulos, E. Nicchiarelli, and R. Sebastiani. Reasoning with Goal Models. In *Proc. of ER 2002*, pages 167–181, 2002.
10. M. L. Griss, J. M. Favaro, and M. d'Alessandro. Integrating Feature Modeling with the RSEB. In *Proc. of ICSR'98*, pages 76 –85, jun. 1998.
11. K. Henricksen and J. Indulska. Developing Context-aware Pervasive Computing Applications: Models and Approach. In *Pervasive and Mobile Computing*, volume 2, pages 37–64, 2004.
12. A. Lapouchnian, Y. Yu, S. Liaskos, and J. Mylopoulos. Requirements-driven Design of Autonomic Application Software. In *Proc. of CASCON 2006*, 2006.
13. J. Mylopoulos, L. Chung, and B. Nixon. Representing and Using Nonfunctional Requirements: A Process-Oriented Approach. *IEEE Transactions on Software Engineering*, 18(6):483–497, 1992.
14. F. Paternò. ConcurTaskTrees: An Engineered Approach to Model-based Design of Interactive Systems. In *The Handbook of Analysis for Human-Computer Interaction*, pages 483–500. Lawrence Erlbaum Associates, 2002.
15. C. J. Pavlovski and J. Zou. Non-Functional Requirements in Business Process Modeling. In *Proc. of APCCM 08*, page 103112, 2008.
16. E. Santos, J. Pimentel, D. Dermeval, J. Castro, and O. Pastor. Using NFR and Context to Deal with Adaptability in Business Process Models. In *Proc. of RE@RunTime*, 2011.
17. M. Satyanarayanan. Pervasive computing: Vision and challenges. *Personal Communications, IEEE*, 8(4):10–17, 2001.
18. E. Serral, P. Valderas, and V. Pelechano. Supporting Runtime System Evolution to Adapt to User Behaviour. In *Proc. of CAiSE'10*, pages 378–392, 2010.
19. D. Shepard. A Two-dimensional Interpolation Function for Irregularly-spaced Data. In *Proc. of the ACM national conference*, pages 517–524, 1968.
20. A. Shepherd. *Hierarchical Task Analysis.* Taylor & Francis, London, 2001.
21. V. E. Silva Souza, A. Lapouchnian, W. N. Robinson, and J. Mylopoulos. Awareness Requirements for Adaptive Systems. In *Proc. of SEAMS '11*, pages 60–69, 2011.
22. A. Sutcliffe, S. Fickas, and M.M. Sohlberg. Personal and Contextual Requirements Engineering. In *Proc. of RE 2005*, pages 19–28, 2005.
23. E. Yu. *Modelling Strategies Relationships for Process Reengineering.* PhD thesis, Department of computer science, University of Toronto, 1995.