Semantic Modelling in Support of Adaptive Multimodal Interface Design

Elena Tsiporkova, Anna Hristoskova, Tom Tourwé, Tom Stevens

Semantic Modelling in Support of Adaptive Multimodal Interface Design

Elena Tsiporkova[1], Anna Hristoskova[2], Tom Tourwé[1] and Tom Stevens[3]

[1]Sirris – Software Engineering & ICT Group
A. Reyerslaan 80 – 1030 Brussels – Belgium
{elena.tsiporkova,tom.tourwe}@sirris.be
[2] Department of Information Technology, Ghent University – iMinds
Gaston Crommenlaan 8 (201) – 9050 Ghent – Belgium
anna.hristoskova@intec.UGent.be
[3]Namahn – Agency for human-centered design
Grensstraat 21 – 1210 Brussels – Belgium
ts@namahn.com

Abstract. The design of multimodal interfaces requires intelligent data interpretation in order to guarantee seamless adaptation to the user's needs and context. HMI (human-machine interaction) design accommodates varying forms of interaction patterns, depending on what is most appropriate for a particular user at a particular time. These design patterns are a powerful means of documenting reusable design know-how. The semantic modelling framework in this paper captures the available domain knowledge in the field of multimodal interface design and supports adaptive HMIs. A collection of multimodal design patterns is constructed from a diversity of real-world applications and organized into a meaningful repository. This enables a uniform and unambiguous description easing their identification, comprehensibility and applicability.
Keywords: Human-machine interface, Multimodal Design Patterns, Adaptive Interfaces, Pro-active Interaction, Data Modelling, Context-awareness.

1 Introduction

The design of *multimodal* and *adaptive* interfaces for complex real-time applications requires a specific approach in order to guarantee that the interaction between human and computer remains natural [1]. In order for the interface to adapt to the user and the context, the system needs to reason about his/her needs and proactively adapt to these while keeping the user in control. Thus, the HMI (human-machine interaction) design needs to accommodate varying forms of interaction, depending on what is most appropriate for that particular user at that particular time.

The contribution described in this paper has been developed in the context of ASTUTE[1], a large EU research project. The project focuses on the design of intelligent multimodal user interfaces, providing pro-active decision support to the

[1] ASTUTE Pro-active decision support for data-intensive environments; http://astute-project.eu/

user. The goal is to develop a platform for building embedded products that capture, reason and act upon user intentions thereby taking into account his/her context (i.e. environment and all the factors which influence his/her performance) and state (i.e. aspects determining his/her ability to perform in a given situation, such as stress level, fatigue). The project approach is validated by various industrial demonstrators in the domains of automotive, avionics, emergency dispatching, building management and manufacturing process management. One of the goals of the project is to provide an appropriate methodology and tools for user interface design, based on design patterns.

HMI design patterns play a major role in exploring and specifying the interactions between users and devices by inspiring design and enabling the reuse of concrete solutions through formal descriptions [2]. An abundance of resources collect design patterns such as Yahoo! Design Pattern Library [3], Patterny [4], search patterns [5], gesture interfaces [6], rich interaction design principles on the Web [7, 8], patterns for effective interaction design [9]. The research toward the establishment of formal principles and guidelines for multimodal interaction design is also gaining increasing interest and importance in recent years (e.g. [10-15]). However, as observed by Sarter [16], the existing guidelines mostly focus on high-level design objectives and do not provide support on how to map them to the needs of an actual application.

In order to facilitate the design of multimodal interfaces in practice, this article pursues the definition of a semantic modelling framework, which has the capacity to: 1) capture and model design patterns[2], formal guidelines and expert domain knowledge in the field of multimodal interface design; 2) reason and derive explicit recommendations in support of both the HMI design and the dynamic HMI adaptation at runtime. A collection of multimodal design patterns exploring diversity of real-world applications is constructed and organized into a meaningful pattern repository. This repository is complemented and extended with ontologies defining relevant use case specific knowledge. Two application examples of the described patterns for the case of emergency dispatching and manufacturing process management are discussed.

2 Semantic Modelling Framework for Design Patterns

2.1 Design Pattern Parameterisation

A substantial part of the work has been devoted to the development of a formal parameterisation framework allowing for the construction of an HMI design pattern repository. A set of parameters for HMI pattern description and specification has been derived through multiple interactive and iterative sessions between a mixed team of HMI designers and ontology engineers. The goal was to develop a uniform HMI pattern model reflecting the need for a formalized description and an increased level of abstraction detail. This supports the decision of pattern applicability in a certain design context. In the spirit of ontology-based modelling, the resulting model is a hierarchical class structure, describing pattern parameters and their attributes [18]:

[2] While respecting the description structure as proposed by the Gang of Four in [17].

- *Pattern Parameters*: User (Junior, Senior), Environment, Device, Task (Primary, Secondary), Interaction, Information, Modality (Input, Output), Event, System.
- *Parameter Attributes*: User Attributes (Expertise, Role, State), Environment Attributes (Safety Level, Change Rate, Noise Level), Interaction Attributes (Interaction Pattern, Interaction Type), Device Attributes (Component, Size).

Various relationships have been derived between the parameter/attribute classes and subclasses as demonstrated later in Section 3.3.

2.2 Layers of Modelling Abstraction

The proposed hierarchical parameter model and the relationships defined between the different parameter classes have been modelled as an ontology enabling further model refinement and reasoning. The method structures the ontological model in a nested set of models consisting of three levels of abstraction as presented in Fig. 1:

- **Core Design Pattern Model**: Complex ontological model containing the generic key concepts and knowledge relevant to design patterns which creation and maintenance requires solid expertise in ontology modelling. HMI designers are closely involved as a source of domain knowledge and for validation purposes.
- **Repository of Design Patterns**: A collection of relatively simple ontological models, defining the specific properties and applicability of each pattern. The main challenge lies in mapping the natural language description of a pattern into a formal logic representation without losing too much expressiveness. This requires close collaboration between ontology experts and HMI designers.
- **Use Case Specific Models:** The purpose of this level is twofold: 1) to derive design recommendations (e.g. applicable design patterns) during the interface design; 2) to allow dynamic interface adaptation (e.g. context-aware output modality) at runtime. The concrete use case is described by making use of the concepts and relationships defined in the design model. HMI designers can execute this autonomously if a suitable (e.g. web-based) interface is provided.

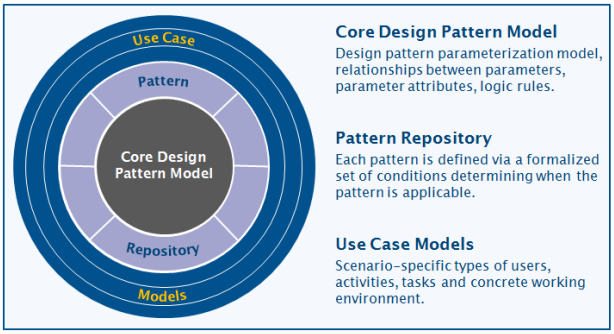

Fig. 1. Schematic overview of the different levels of semantic modelling abstraction.

The core design pattern ontology model consists of 2 types of high-level key domain concepts *Parameter* and *ParameterAttribute*, describing the design pattern parameters and their attributes. These are related via the hasAttribute relationship.

Each parameter is connected with its attributes through a set of relationships, which are sub-properties of this property. Some concrete examples are given in Table 1.

Table 1. Relation between specific *Parameters* and their *Attributes.*

Parameter	**Relation**	**Attribute**	**Description**
User	hasExpertise	*Expertise*	expertise level of the application user, e.g. *first_time_user* or *advanced*
	hasAttention	*Attention*	focus of the user's attention, e.g. *environment*, *on_single_device*, *distributed_among_devices*
	canInteractVia	*InteractionChannel*	interaction capacity of the user, e.g. *hearing*, *voice*, *touch*, *sight*
Device	hasComponent	*Component*	device components e.g. *Video* (*camera*, *display*), *Haptic* (*vibration_device*)
Task	hasStructure	*Structure*	user task consists of a *single_action* or a *composition_of_actions*
Interaction	exhibitsPattern	*InteractionPattern*	applicable design pattern for the concrete situation, e.g. *important_message_pattern*, *adapting_context_detail_pattern*, *combination_of_modalities_pattern* (detailed in Section 5)
Environment	hasChangeRate	*ChangeRate*	frequency of change occurring in the environment, e.g. *frequent*, *never*, *rare*
	hasSafetyLevel	*SafetyLevel*	safety level of the environment, e.g. *potential_risk*, *risk_full*

The core ontology also defines a multitude of relationships between the different *Parameter* subclasses. Several concrete examples are visualized in Fig. 2.

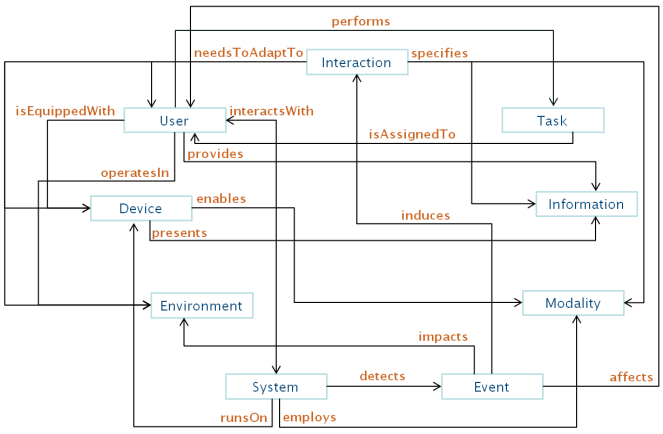

Fig. 2. Some of the relationships between the subclasses of the *Parameter* class.

In addition to the class hierarchy and relationships defined between the different subclasses of the hierarchy, the core ontology is also equipped with some reasoning capacity. Two types of reasoning rules are implemented:

- Chain rules allowing to deduce a relationship based on the transitive application of two other relationships, e.g.: *(Interaction* needsToAdaptTo *User)* **AND** *(User* isEquippedWith *Device)* => (*Interaction* needsToAdaptTo *Device)*
- SWRL rules expressing some common sense knowledge or logic rules applicable to any Design Pattern, e.g.: **IF** (*System* detects *Event*) **AND** (*Event* hasCriticallityLevel *severe*) **THEN** (*Event* hasPriority *high*)

The original core ontology is complemented and extended by creating a separate ontology for each design pattern. Such an ontology needs to 1) import the Core Design Pattern Model ontology; 2) instantiate the *InteractionPattern* class with the concrete Design Pattern; 3) describe in a formal fashion (using SWRL rules) the conditions under which the concrete Design Pattern is applicable, e.g. Table 2:

Table 2. Examples of SWRL description of the applicability conditions for design patterns.

Patterns	**Applicability Conditions**
Important Message	**IF** (*System* detects *Event*) **AND** (*Event* hasPriority *high*) **AND** (*User* hasAttention *on_environment*) **AND** (*Event* induces *Interaction*) **THEN** (*Interaction* exhibitsPattern *important_message_pattern*)
Combination of Modalities	**IF** (*System* detects *Event*) **AND** (*Event* impacts *Environment*) **AND** (*Environment* hasSafetyLevel *risk_full*) **AND** (*System* sends *Information*) **AND** (*Information* hasType *instruction*) **AND** (*Event* induces *Interaction*) **THEN** (*Interaction* exhibitsPattern *combination_of_modalities_pattern*)
Adapting Context Detail	**IF** (*System* detects *Event*) **AND** (*System* displays *Information*) **AND** (*Information* hasType *status_update*) **AND** (*User* canInteractVia *sight*) **AND** (*User* hasGoal *UserGoal*) **AND** (*Event* increasesProximityTo *UserGoal*) **AND** (*Event* induces *Interaction*) **THEN** (*Interaction* exhibitsPattern *adapting_context_detail_pattern*)

3 Use Case Models

This section provides two application examples of the described patterns for the case of emergency dispatching and manufacturing process management. The original core ontology and the repository of design patterns are complemented and extended by creating ontologies with relevant use case specific knowledge.

3.1 Emergency Dispatching Management Application

A fire commander instructs his firefighters to evacuate the building that is on fire. The ontologies discussed above are instantiated with information such as: 1) *types of users involved*: fire fighters, fire team commanders, fire station dispatchers, air sampling collectors, medical experts; 2) *activities and tasks*: evacuation, search and rescue, locating water supplies, search and rescue, defining security perimeters in the presence of dangerous substances; 3) *applications and devices:* device type (mobile, sensor), status (active, idle), components (audio, video, haptic), application features (supported users, detected events); 4) *working environment:* inside/outside, noise level, security level; 5) *events*: type (toxic smoke formation, approaching dangerous goods), priority, criticality level.

The purpose of this level is to recommend design patterns and derive new knowledge from the models and data in the different abstraction layers. For instance, an ontology describing the emergency dispatching case may contain the following:

Parameter	**Relation**	**Attribute**
fire_commander	performs	*coordination_of_evacuation*
	operatesIn	*site_on_fire*
	isEquippedWith	*fire_commander_tablet*
coordination_of_evacuation	isFocusedOn	*site_on_fire*
	isPartOf	*coordination_of_emergency*
fire_commander_tablet	hasComponent	*loudspeaker, touch_screen*
	hasStatus	*active*
	isComplementedBy	*fire_commander_smart_phone*
toxic_smoke_formation_app	detects	*toxic_smoke_formation_event*
toxic_smoke_formation_event	hasCriticalityLevel	*severe*
	impacts	*site_on_fire*
	affects	*fire_commander*
	induces	*toxic_smoke_formation_interaction*
SWRL rules	**IF** *(System detects toxic_smoke_formation_event)* **THEN** (*System* sends *toxic_smoke_formation_message*)	

Then the following additional information and recommendations will be derived:

Parameter	**Relation**	**Attribute**
toxic_smoke_formation_app	interactsWith	*fire_commander*
	informs	*fire_commander*
toxic_smoke_formation_event	hasPriority	*high*
toxic_smoke_formation_interaction	exhibitsPattern	*important_message_pattern*
	specifies	*body_awareness, visual_out*
toxic_smoke_formation_message	isPresentedOn	*fire_commander_tablet*
	hasType	*alarm*

3.2 Manufacturing Process Management Application

A team leader monitors a production line. By approaching it the system zooms in and offers richer information about its productivity, such as the Overall Equipment Efficiency (OEE), which defines the quality, performance and availability of the line. The design pattern ontologies are instantiated with the following information: *1) types of users involved*: team leaders, production line workers, maintenance worker; 2) *activities and tasks*: monitoring of and working on the production line, repairing machines; 3) *applications and devices:* device type (mobile, sensor), status (active, idle), components (audio, video, haptic, display, keyboard), application features (users supported, events detected); 4) *working environment:* production line space, administration office, visibility level; 5) *events*: approaching production line.

The production management ontology contains the following data for deriving design recommendations and knowledge from the abstraction layers' models:

Parameter	Relation	Attribute
team_leader	performs	*monitors_production_line*
	isEquippedWith	*team_leader_tablet*
	canInteractVia	*sight*
monitors_production_line	isFocusedOn	*production_line*
	requires	*production_line_overview_screen*
	isAssignedTo	*team_leader*
production_line	hasVisibilityQuality	*excellent*
team_leader_tablet	hasComponent	*loudspeaker, touch_screen*
	hasStatus	*active*
approach_production_line_app	detects	*approach_production_line_event*
approach_production_line_event	hasCriticalityLevel	*moderate*
	impacts	*production_line*
	affects	*team_leader*
	induces	*approach_production_line_interaction*
	increasesProximityTo	*production_line*
production_line_overview_screen	hasType	*status_update*
	isAbout	*overall_equipment_efficiency_oee*
SWRL rules	**IF** *(System* detects *approach_production_line_event)* **THEN** *(System* sends *production_line_context_details)*	

Consequently, the following information and recommendations are derived:

Parameter	Relation	Attribute
approach_production_line_app	interactsWith	*team_leader*
	employs	*visual_out*
	runsOn	*team_leader_tablet*
	operatesIn	*production_line*
	sends	*production_line_context_details*
approach_production_line_interaction	specifies	*production_line_context_details*
	exhibitsPattern	*adapting_context_detail_pattern*
	needsToAdaptTo	*team_leader, team_leader_tablet*
production_line_overview_screen	isPresentedOn	*team_leader_tablet*
	hasDetailGranularity	*detailed*

4 Conclusion

This paper presents a semantic technology approach to formally model and exploit relevant domain knowledge about HMI design patterns. Our aim is to demonstrate that the modelling of appropriate data about user state and context linked to the specification of interaction patterns constitute a powerful means to realise a proactive, adaptive multi-modal interaction both at design time and runtime. Besides further refinement of the presented semantic modelling framework, we intend to open our pattern repository to the benefit of the HMI community and thus enable different

stakeholders in the design domain to interact, exchange ideas, improve and annotate patterns proposed by others, contribute new patterns and describe other use cases.

References

1. Plass-Oude Bos, D., Poel, M., Nijholt, A.: A Study in User-Centered Design and Evaluation of Mental Tasks for BCI. Lecture Notes in Computer Science, Advances in Multimedia Modeling, 6524, pp. 122—134 (2011).
2. Goodwin, K.: Designing for the digital age: How to create human-centered products and services. Wiley (2011).
3. Yahoo! Design Pattern Library, [Online]. Available: http://developer.yahoo.com/ypatterns/.
4. Oy, P. F.: Patterny, Pattern Factory Oy (Ltd.), [Online]. Available: http://patternry.com/.
5. Morville, P.: Callender, J.: Search Patterns: Design for Discovery, O'Reilly Media Inc (2010).
6. Saffer, D.: Designing Gestural interfaces, Sebastopol, O'Reilly Media Inc (2009).
7. Scott, B., Neil, T.: Designing Web Interfaces, Principles and Patterns for Rich Interaction. O'Reilly Media Inc (2009).
8. Welie, M. V.: Patterns in interaction design - Patterns library. 2008. [Online]. Available: http://www.welie.com/.
9. Tidwell, J.: Designing interfaces, Patterns for effective interaction design, O'Reilly Media Inc (2010).
10. Godet-Bar, G. , Dupuy-Chessa, S. , Nigay, L.: Towards a system of Patterns for the design of Multimodal Interfaces. Computer-Aided Design of User Interfaces V, Proceedings of the Sixth International Conference on Computer-Aided Design of User Interfaces CADUI '06, Bucharest (2006).
11. Ratzka, A., Wolff, C.: A Pattern-Based Methodology for Multimodal Interaction Design. Lecture Notes in Computer Science, 4188, pp. 677-686 (2006).
12. Ratzka, A.: Identifying User Interface Patterns from Pertinent Multimodal Interaction Use Cases. Mensch und Computer 2008: Viel Mehr Interaktion (2008).
13. Dumas, B., Lalanne, D., Oviatt, S.: Multimodal interfaces: A survey of principles, models and frameworks. Human Machine Interaction, pp. 3-26 (2009).
14. Oviatt, S.: Multimodal interfaces. The human-computer interaction handbook: Fundamentals, evolving technologies and emerging applications, pp. 286-304 (2003)
15. Reeves, L., Lai, J., Larson, J., Oviatt, S., Balaji, T., Buisine, S., Collings, P., Cohen, P., Kraal, B., Martin, J., et al.: Guidelines for multimodal user interface design. Communications of the ACM 47(1), pp. 57-59 (2004)
16. Sarter, N.: Multimodal information presentation: Design guidance and research challenges. International journal of industrial ergonomics 36(5), pp. 439-445 (2006)
17. Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design patterns: elements of reusable object-oriented software, Addison-Wesley (1994)
18. Ferreira, N., Geldof, S., Stevens, T., Tourwé, T., Tsiporkova E.: Patterns for HMI design of multi-modal, real-time, proactive systems. In: Proceedings of the IUI 2013 Workshop on Interacting with Smart Objects (2013)