Multi-Dimensional Context-Aware Adaptation of Service Front-Ends

Project no. FP7 – ICT – 258030

Deliverable 4.2.2
Algorithms for Advanced Adaptation Logic

Due date of deliverable: 31/08/2012

Actual submission to EC date: 31/08/2012

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)

Dissemination level		
[PU]	[Pubic]	Yes

Document Information	
Lead Contractor	UCL
Editor	Vivian Genaro Motti, Nesrine Mezhoudi, Jean Vanderdonckt
Revision	22.08.2012
Reviewer 1	Dave Raggett
Reviewer 2	
Approved by	
Project Officer	Michel Lacroix

Contributors	
Partner	**Contributors**
UCL	**Vivian Motti, Nesrine Mezhoudi, Jean Vanderdonckt, Pascal Beaujeant, Sophie Dupuis**

Changes			
Version	**Date**	**Author**	**Comments**
1	04/05/2012	UCL	First structure of the documents, and descriptions.
2	21/06/2012	UCL	Refinement of structure, draft of the contents

Executive Summary

This deliverable is the second release of the advanced adaptation logic (D4.2.1) of Serenoa project. In the first release we dedicated our efforts on gathering several algorithms that adapt different resource types, as text, images, and videos. In the second release we investigated how machine learning algorithms can be effectively applied to perform more complex adaptation logics, i.e. by combining several specific algorithms, optimizing the inference process, and selecting more concrete adaptation scenarios.

Once the outcome of this task is mainly in a prototype format, we describe the decisions taken and the characteristics of the algorithms implemented.

Table of Contents

1 Introduction

1.1 Objectives

This deliverable presents and describes possible applications of machine learning algorithms in order to perform advanced adaptation logic. In this sense, two application scenarios were initially defined and then related algorithms were selected, implemented, and discussed.

1.2 Audience

The target audience for this deliverable is composed by researchers and practitioners from both scientific and industrial domains with interest in the topic of context-aware adaptation and machine learning techniques.

1.3 Related documents

- D4.2.1 – Algorithms for Advanced Adaptation Logic: the first release of this document provides the list of specific algorithms that perform context-aware adaptation
- D2.4.2 - Evaluation Criteria: defines quality metrics that are relevant for analysing benefits of these algorithms
- D5.2.2 presents prototypes and scenarios that can benefit from AAL (advanced adaptation logic)

1.4 Organization of this document

Chapter 1 presents the goal, audience and related documentation with this deliverable. In Chapter 2, the theoretical background and main definitions are provided. In Chapter 3 the first scenario of application and respective design of the algorithms are explained. In Chapter 4 the second scenario of application is presented, and relevant algorithms are described. In Chapter 5 a discussion and conclusion are presented.

2 Fundamental Concepts

Machine Learning (ML) algorithms can be applied to solve problems and support decisions in several different application domains. In the domain of Human-Computer Interaction (HCI), for instance, such algorithms can be applied to adapt the user interfaces (UI), e.g. by matching context information with appropriate adaptation rules according to the users, based on their profiles or interaction history. By learning with previous experiences acquired within a specific task, these algorithms evolve, improving their performance and accuracy, and becoming more efficient, for instance while classifying data or recommending contents.

When applied to the HCI domain, ML can support context-aware adaptation (CAA), mainly by identifying the context of use, handling its complexity to decide the more appropriate changes, and finally adapting the system in response to specific events or context information.

There are several ML algorithms that are capable of supporting CAA in its different phases and scenarios. Clustering can be used to identify relationships among context information, regression can be used to associate evaluation criteria, classification can be used to group contexts of use, and decision trees can support the selection of adaptation rules.

2.1 Techniques

In this deliverable we explain the application, and analyse the following algorithms:

- Regression
- Clustering
- Decision Trees

2.1.1 Regression

Definition. Regression is a technique to model and analyse data, mainly used to identify the relationship between a dependent and an independent variable. Thus, the analysis of such relationship helps to understand the variation of one value according to another. Regression analysis is also used to understand which independent variables are related to the dependent one.

One of the main applications of regression is prediction, overlapping with machine learning domain. The goal of applying regression analysis is to develop a model that can be used to predict similar associations for future experiments. In the CAA context such a model could guide developers in selecting more appropriate adaptations, for instance according to quality metrics (e.g. ergonomics).

Modelling. Examples of dimensions that can be considered for the regression in the domain of advanced adaptation logic include:

- the number of widgets (pondered by their respective dimensions)
- the number of colors, shapes, widget types
- screen dimensions (weight, height, diagonal)

A linear regression can be modelled to identify the following knowledge (taking into account a sample of UIs with high ratings of usability and ergonomics):

- Is there a correlation between ergonomic metrics and specific dimensions?
- Is there a regression equation that can be taken into account to guide the adaptation process? E.g. classifying samples as over (or under) aligned, balanced, dense, etc?

Application. Application scenarios include: (i) the implementation of an automatic evaluation tool (by taking a UI as an input, analysing its characteristics, extracting parameters of interest, calculating its position in the graphic and defining the evaluation results), the regression can be used also to support the decision of which techniques for adaptation are the most appropriate ones; (ii) recognizing the relationship between UI figures (e.g. number of widgets, size) and context information (e.g.: screen size, aesthetic metrics); (iii) associating quality measures (e.g. ergonomics, aesthetics) and UI design (e.g. widget sizes).

Such scenarios can benefit of taking a large sample of UIs and analysing them regarding the pre-defined

criteria. E.g. the relationship between balance and the amount and dimensions of UI elements is useful to guide the adaptation of UI elements.

2.1.2 Clustering

Definition. Clustering has been largely applied as a technique for data mining and statistical data analysis to support pattern recognition, image analysis and information retrieval. It consists in grouping instances in clusters (groups) according to their similar characteristics.

Modelling. To implement a clustering algorithm the following steps must be respected:

- each instance (e.g. user) must be described in a model containing its characteristics (e.g. age, gender, preferences, etc);
- each instance must be associated with a certain weight (calculated based on the model);
- each weight is associated with a normalised descriptor (e.g. 1, 2 or 3)
- then, a filtering algorithm may be applied
    - first calculating the distance between each pair of instances (e.g. Fowlkes-Mallows index[1] or Jaccard index[2]), quantifying the similarities between two datasets;
    - then selecting similar profiles;
    - and finally, predicting relevant concepts according to the similarities identified.

Application. Based on selected properties of different instances, clustering identifies a standard profile that connects them. To model platforms or users for examples, classes of instances can be found and applied to support the CAA, e.g. for collaborative filtering algorithms. When user profiles are compared, their similarity can be extracted according to their behaviour or preferences, enabling the system to make relevant recommendations based on similar profiles.

2.1.3 Decision Trees

Definition. A decision tree is a way of representing and classifying data. It resembles the tree structure consisting of a set of nodes and a set of directed edges that connect the nodes, as a directed acyclic graph. The internal nodes stand for questions (or conditions) that can be evaluated, the edges correspond to answers to those questions, and the leaf nodes represent the final answer – also called decision [Gomez, 2009].

Modelling. In general, two steps are necessary for modelling a decision tree: the conditions that will be evaluated in each node must be defined (in an association involving a given instance, an operator, and a specific value), then the leaves of the tree must be defined, usually in terms of classes (for classification tasks) or in terms of actions (for a recommendation scenario). The conditions, or nodes, can be manually defined or vary dynamically according to the context of use, application domain and training with real data.

Application. In the domain of HCI, and more specifically of context-aware adaptation, decision trees can be applied to suggest relevant contents, appropriate UI elements, or pertinent adaptation techniques in a given scenario.

2.2 Scenarios

2.2.1 Justification

The main benefits of applying machine learning consist in taking intelligent decisions based on defined examples. Such examples can be used to find characteristics of interest (discovering), for instance by recognizing potential associations or patterns that are useful for predicting something.

2.2.2 Learning

In the domain of artificial intelligence, the learning process occurs when the system is able to analyse data in order to either abstract some concepts or to find patterns. Learning goals consist in an attempt to optimize or

[1] Fowlkes and Mallows (1983) proposed a method for comparing clusters
[2] The Jaccard index, or similarity coefficient, is defined by the size of an intersection between partitions A and B, divided by the size of the union of the sample sets

simplify the processing or the interaction. When applied to the domain of HCI a system can learn based on the history of the user (e.g. when a specific set of tasks is always executed in the same order, a system can 'learn' the standard path of the interaction of the user), or based on the preferences of the user (e.g. when the end users always configure the same setting for their interaction, the system can 'learn' their preferences, for instance regarding the volume of an audio content or the preferred brightness level of the screen).

Learning can be a *supervisioned* or a *non-supervisioned* activity:

- In **supervisioned** learning, there is a set of data previously gathered, that is available for the system to analyse, the so-called training examples. By analysing such examples the system is able to perform classification or regression. According to the two examples mentioned above, training data consists in the history of interaction of several users for a given set of tasks, or the configuration settings of several users. With the training examples, the system is able to look for and identify some patterns and co-relations, if they exist.
- In **non-supervisioned** learning, there is no parameter to evaluate the potential solution, the techniques involved try to summarize and explain data, for instance by finding groups of end users with similar profiles or behaviours (Clustering).

Regardless of the activity type, a machine learning algorithm can also have the support and collaboration of a human, i.e. instead of performing all the data analysis alone, the human intuition can be applied to better adjust the performance of an algorithm.

2.3 Template

In order to obtain a unified description of algorithms presented in this deliverable the following template was defined:

Algorithms presentation: provides a brief description about the algorithm

Advantages: provides a list of strong points of the algorithm

Disadvantages: defines the disadvantages of such algorithm

```
Algorithm #: Algorithm_name

    Input:
    /* comments */

    Output:

Begin
    /*pseudo-code*/
End
```

The template includes the name of the algorithm, required input, expected outputs as well as the pseudo-code itself.

3 Scenario 1: Task model to AUI model

3.1 Formal Definition of the Problem

This application scenario consists in applying machine learning algorithms to help to support the decisions taken while transforming a task model into an abstract user interface (AUI) model, within a given context of use and application domain.

3.2 Related Works

Several works have been applied in the same context and they were reported in the literature. For instance:

- Roam Transformation Manager (by Chu et al., 2003)
- Dygimes (by Coninx et al., 2003)
- MOBI-D (Puerta and Maulsby, 1997)
- CTT to Graph

3.3 Algorithms

In this section we describe in details the algorithms mentioned above, they were classified according to the possibility or not of the user to intervene in the algorithm.

3.3.1 Roam

Algorithms presentation: ROAM enables device independent components to be transformed at runtime to fit the target device capabilities. It provides a device independent GUI toolkit that a developer can use to build user interface (UI) components. At migration time or runtime, these device-independent UI components are transformed to run on the target device.

Reference: Chu et al., 2003

Advantages: ROAM application show that SGUI toolkit can generate consistent presentations across different platforms in term of three aspects of consistencies: task consistency, layout consistency, and transformation consistency

Disadvantages: It is difficult to customize a device independent representation for a particular device. It does not support migration for real-time applications: a real-time constraint on migration latency.

Algorithm 01: Roam Transformation Manager

```
      Input:  tt:task tree (CTT) , P:platform
      Output: UI components
            Local variable: v:node
Begin
vlist:= Lowest-level-unsplittbal-node(tt)
foreach v in vlist do
widget-list:= styl-app (v.child(),P)
Apply-grid-bag-layout( widget-list)
PS=page-size-cacul()
if PS > P.screensize()   /*over-filled page treatment */
     Apply-flow-layout()
```

```
        PS=page-size-cacul()
      if (PS > P.screensize() & v.is-splittable())then
         allocate-new-page()
     else
         if (PS > P.screensize() & not(v.is-splittable()))then
          widget-list :=trasform-widget(v,P)
          Apply-flow-layout(widget-list)
          PS=page-size-cacul()
         else
             if (PS > P.screensize())then
            delete-overfitt-widgets()
            end
        end
     end
else    /*under-filled page  treatment */
repeat
v:= parent(v)
widget-list.add (child(v),P)
Apply-grid-bag-layout( widget-list)
PS=page-size-cacul()
Untill (PS  > P.screensize())
End
If  (PS = P.screensize())then
allocate-new-page()
end

End
```

3.3.2 Dygimes

Algorithms presentation: Dygimes is a framework for dynamically generating User Interfaces for embedded systems and mobile computing devices. Runtime transformations of UIs for adaptation to the target device are also supported. Dygimes supports different methods to carry out the specified interactions. The Direct Method Invocation (DMI) is used, ensuring performance, and combined with the use of web service messaging protocols enabling Remote Procedure Calls (RPC) in an XML-syntax to invoke application functionalities.

Reference: Coninx et al., 2003

Advantages: The system eases the creation of consistent, reusable and easy migratable UIs. The UIs can automatically adapt to new devices, offering the same functionality, without being redesigned. The Dygimes framework is already successfully used in the SEESCOA project.

Disadvantages: The system does not support multiple or mixed modalities, and can not migrate from one device to another device.

Algorithm 02: Dygimes

```
Input: CTT : annotated task model
Output: generated UIs

Begin
      Calculate-ETS (ctt);  // content of dialog windows
      Define-STN ();// provide navigation
      ETS-verification ()// designer review
End
```

3.3.3 MOBI-D

Algorithms presentation: End users can describe the set of tasks, then formal models are built based on users' tasks and domain objects. The system supports the development of presentation and dialog specification, based on the models, enabling the visualization of interface designs.

Reference: Puerta and Maulsby, 1997

Advantages: The algorithm allows the definition of the different possible abstract user interface according to human factors. It aims at optimizing the structure of abstract containers. The decision support mechanisms in MOBI-D use the user-task and domain models to make recommendations for presentation and interaction techniques.

Disadvantages: Given that the algorithm was defined in 1997, period of a different technological landscape, it is possible that some of its concepts are obsolete, as such updates are needed, for instance regarding the UI elements and contexts of use were not previously considered

Algorithm 03: MOBI-D

```
Input: task tree,
Output: recommendations for UI elements, models
Begin
      Terms = Parsing(task_tree, key_objects, key_actions)
      For each (term){
            Edit (term)
            Refine (term)
            }
      Generate(structured_task_description,term)
      Generate(user_task)
      Generate(domain_mpdel)
      Integrate(models)
End
```

3.3.4 TOMATO-L

Algorithms presentation: AUI generation based on graph theory and searching methods, the tasks properties (order and grouping) are defined according to their type, weight, and also according to the context (platform description and users' profile)

Reference: Limbourg and Vanderdonckt, 2004

Advantages: The algorithm allows the definition of the different possible abstract user interface according to human factors, it is flexible, reusable, and covers a general-purpose. It aims at identifying an optimal abstract container partition. The proposed algorithm differs in term of considered constraints and the way of exploring them. In the discussion section we give more details about constraints and their uses.

Disadvantages: some abstraction effort is required by the person who is responsible to incorporate the design knowledge.

Algorithm 05: TOMATO-L (by Limbourg and Vanderdonckt, 2004)

```
Input: ctt: a taks tree, dm: domain model, V : root task, PM :
platform model

Output: abstract UI

Begin

/*analyse the task tree, resulting in a semi ordered list
according to tasks hierarchy, their type and weight*/

TasksFlow:= TreeTraverse(ctt,V)

/*define container based on platform description and human factor
as criteria*/

S:=Split( tasksFlow, PM)

/*in order to explore the neighbordhood space to find more
solutions several strategies can be adopted (e.g. merging, drop-
add, add-Drop, oscillation), to decide one, an evaluation step is
applied, according to the constraints defined to select feasible
solutions */

neighbordshipExploration(S);

End
```

3.4 Discussion

This sections presented a set of algorithms dedicated to support the transformation between task trees and abstract UI models. As a taxonomy for possible tasks that are considered, we highlight: convey, create, reinitialize, filter, delete, duplicate, navigate, perceive, move, modify, mediate, select, trigger, stop, and toggle. Then, as possible metrics that are relevant to weight such tasks, we mainly consider: their workload (measure in terms of the physical and mental requirements associated), cognitive (mental efforts), computational (in terms of resources and processing), physical space (dimensions of the UI), time (on average that an end user needs to conclude to the task), and resource consumption (e.g. requests needed).

In order to measure the workload, Nasa-TLX defines important factors that directly impact it. For instance, the *mental demand* is characterized by the mental and perceptual activities that are required from the end user (such as: thinking, deciding, calculating, remembering, looking, and searching). Their complexity and easiness are also involved. The *physical demand* refers to the requirements in terms of physical activities, such as: pushing, pulling, turning, controlling or activating. The *temporal demand* refers to the need of time that is required. The *performance* refers to the success and satisfaction of the end user in accomplishing his

or her tasks. The *effort* refers to how difficult it is for the end user to conclude his or her tasks. And the *frustration level* refers to the users' feelings of: insecurity, discouragement, irritation, stress and annoyance.

The workload of a task can be identified and used by the system in a given context of use in order to adapt a task tree, i.e. by calculating and analysing this criteria a system is able to better select a task or a group of tasks that are more appropriate within a specific context of use.

4 Scenario 2: Decision Trees in the Widget Selection

4.1 Formal Definition of the Problem

In this scenario the Concrete User Interface (CUI) model is considered. Machine learning algorithms are applied within this context to help and to support the decision of which widgets are the most appropriate ones in a given context of use based on several parameters. Widgets that have equivalent goals, can be considered for the selection. For instance users in a distracted environment (e.g. while driving) must have a highly efficient interaction (i.e. high performance, quick interaction). Thus the algorithms can help in selecting a widget that requires the minimum time for interaction. Mainly we describe and discuss the application of decision trees to support this activity.

4.2 Related Works

Several works have been reported in this domain. We mainly highlight:

- TRIDENT (by Vanderdonckt and Bodart, 1993)
- TIMM (by Eisenstein & Puerta, 2000)
- TRIAD (by Martinez et al., 2010)
- Leiva (2012)

4.3 Algorithms

In this section we present the algorithms that enable the users to intervene in the processing, usually by accepting, evaluating or rejecting the algorithm's decision. According to Eisenstein and Puerta (2000) there is ample reason to believe that machine learning also benefits from end users' advice.

4.3.1 TRIDENT (by Vanderdonckt and Bodart, 1993)

Algorithms presentation: it provides a decision tree (see Figure 1) that takes into account a broad set of discriminants and represent progress towards automated user interface design. TRIDENT is a set of interactive tools that automatically generates a user interface for interactive applications. It includes an intelligent interaction objects selection based on three different concepts. First, a typology classifies abstract interaction objects to allow a presentation independent selection. Second, guidelines are translated into automatic rules to select abstract interaction objects from both an application data model and a dialog model. Third, these guidelines are encapsulated in a decision tree technique to make the reasoning obvious to the user. This approach guarantees a target environment independent user interface. Once this specified, abstract interaction objects are mapped into concrete interaction objects to produce the observable interface.

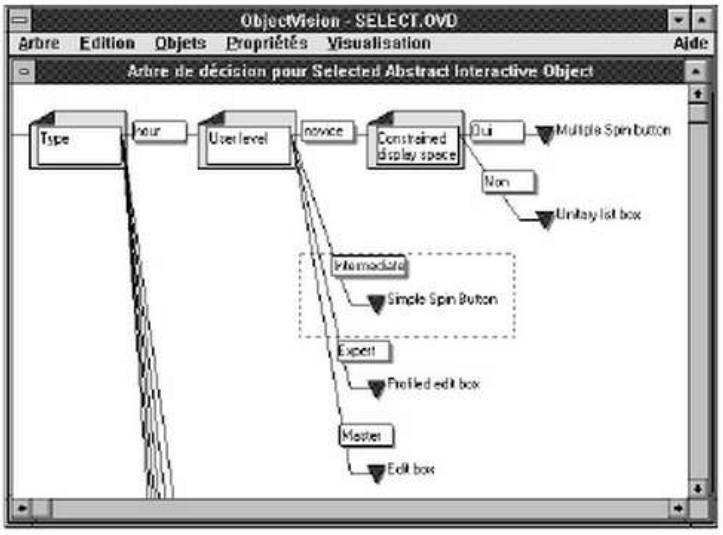

Figure 1. A partial view of the TRIDENT's AIO Selection Tree

Advantages: decision trees are easier to visualize and understand for designers, the user can choose rules and adjust the generation process of the UI, the approach is flexible since the tree can be easily modified, refinements can be performed to reach the best possible solution.

Disadvantages: they do not incorporate adaptation, some identical rules may be duplicated in the tree (at different stages), and the decision tree can become very large depending on the application scenario.

Algorithm 05: TRIDENT

```
Input:
AIO, application data model, dialog model
Output:
User Interface
Begin
Select_abstract_interaction_objects();
For each (AIO_selected) {
      Create_AIO_specification();
      Transform(AIO, CIO);
      }
UI_generation();

End
```

4.3.2 TIMM (by Eisenstein and Puerta, 2000)

Algorithms presentation: a decision tree classify data according to a given criteria (Figure 2). The criteria determine for which class the data will be sorted. Each level of the tree corresponds to a specific criterion, in this case based on the context information. The leaf of the tree defines a recommendation of a widget, based on the selection performed.

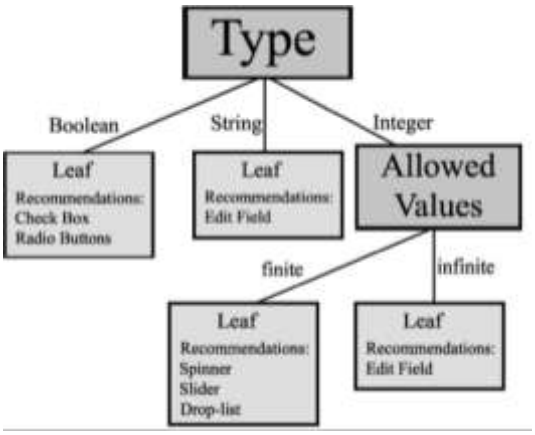

Figure 2. A simple decision tree for Interactor Selection [Eisenstein and Puerta, 2000].

Advantages: Decision trees are easy to read, to understand and to predict its effects. The algorithm is

sensitive to a small number of examples, easy to understand, refined according to the use, and it focuses on important design aspects. The authors believe this algorithm provides significant performance gains and it serves for a general-purpose i.e. can be also applied in other domains.

Disadvantages: Overfitting may occur (when one new rule is created to consider each example of use), affecting the performance of the algorithm. The authors discuss three possible solutions to avoid it: more discriminants, a sensitive threshold or taking the user advice. Some complexity problems can be expected too, since the algorithm searched the entire space of possible changes for the most advantageous alteration in the decision tree. Since the algorithm looks ahead only one step to search for the most beneficial operations, it is likely that a local minima issue may occur.

Algorithm 06: TIMM (by Eisenstein and Puerta)

```
      Input:                                                    A
      set of widgets, context of use
      Output:
      Recommendation about the most appropriate widget within a given
      scenario
Begin
      take_standart_recommendations();
      record_user_selections();
      loop {
            make_recommendations();
            count_errors();
            if (errors==0) break;
            find_best_operation();
            if (error_gain > min_threshold) {
                  apply_operation();
                  next;
            }
            else break;
      }
End
```

4.3.3 TRIAD

Algorithm Presentation: algorithm for generating UIs of different abstraction levels in a RIA (rich internet application) context. TRIAD defines the application as a hierarchy of tasks, in which the leaf nodes are atomic tasks. Temporal operators define task types: sequential, concurrent and choice.

Advantages: This algorithm takes into account contextual information (the platform, temporal relationships between tasks and vicinity of the tasks). The resulted designs could be refined multiple times (different scenarios) in order to explore new configurations. A set of metrics has been proposed and reviewed.

Disadvantages: Metrics seem complex but do not compulsorily ameliorate the process of finding the UI

structure. Besides this, the algorithm do not give optimized solution it just presents a plausible scenario.

Algorithm 07: TRIAD (by Martinez et al., 2010)

```
Input: CTT: task tree
Output: the final UI
Begin
/*recovery of sub tree*/
Repeat
Create-next-Container (anchor, layer)
Until (anchor-node== root)
For each container C do
Apply-generation-schema (subTreeStructure)
Configuration-selection ()
Define-navigation ()
```

4.3.4 Leiva (2012)

Algorithms presentation: Leiva (2012) proposes the re-design of the UI components (widgets), based on the user interaction. Thus the style of the widget is adapted according to the behaviour of the user.

First, a JSON configuration is parsed, then the object keys are used to select marked widgets (e.g. buttons, table cells or text paragraphs). Then, a list of event is created as hash tables to log the interaction data at run time (e.g.: mousemove, click or keydown event lists). Interaction scores values are associated with each widget, and the more the user interacts with it, the higher its score. Adaptations are incrementally applied.

Advantages: this approach is technology-independent (since the data hierarchical structure and style sheets are used), the changes are gradually presented to the end user avoiding a significant disruption.

Disadvantages: only the user interaction is considered as context for the adaptation; only numerical properties of the CSS can be adapted (e.g. dimensions and colors)

Algorithm 08: Leiva (2012)

```
Input: a set of widgets and their properties subjected to
adaptation
Output:
UI with adapted widgets
Begin
Read_and_Parse_JSON_Widgets_Set();
Select(widgets_set);
Track_user_interaction(widgets_set, local_DB);
For each (widget) {
```

```
            Update_score(widget);
            }
      Adapt(widgets,user_interaction);
End
```

4.4 Discussion

As presented in this section, there are some works dedicated to investigate and to apply machine learning algorithms in the selection of widgets. By analysing such works, it is possible to identify concrete scenarios of application, potential widgets for adaptation in this context, and also a set of criteria that are relevant in the selection decision.

As relevant widgets we can highlight: edit fields, scroll bars, combo box, radio box, list box, buttons, radio buttons. The main requirement is that the widget is able to perform the original task. Moreover, as relevant criteria to define the selection, we highlight: the number of values to choose, the total amount of possible values, the nature of the data (e.g.: continuous, discrete, numeric, literal), and the density of the UI,

The most important requirement for the widget selection consists in defining widgets that have the same capabilities, i.e. enable users to perform the same task but in an equivalent manner. Then, for each of the widgets considered, they must be associated with specific weights (e.g. the spatial dimension required in a UI, the time to interact with it, the necessary precision, etc.). Finally, based on these weights of each widget and the context of use (in general), the algorithm is able to support the adaptation decision, by defining the most appropriate element in a given scenario.

Further information, can also be considered to support this decision, for instance quality metrics. Quality metrics (as the ones specified in the D2.4.2 Evaluation Criteria) can also be taken into account in such algorithms, for instance by orienting the selection of widgets according to usability or ergonomics and aiming to achieve better adaptation results.

Moreover, besides guiding the adaptation according to quality metrics, it is also important to enable some sort of collaboration between end users and system, to permit users to intervene in the process, either by accepting, rejecting, or even adapting the adaptation itself, aiming also to improve the results achieved by the adaptation.

5 Final Remarks

This deliverable presented a set of potential algorithms of machine learning that can be successfully applied to adapt the user interface according to the context of use. Each algorithm was illustrated by means of a standard template, its description, advantages and limitations were presented.

In the first release of this deliverable, more specific techniques for adaptation were presented. In this document we focus on how to optimize the use of such techniques by considering further reasoning and inference that is not possible by a simple rule.

5.1 Discussion

Although a limited set of algorithms and scenarios was presented above, we believe that they provide an interesting overview about the current possibilities and scenarios for optimizing the context-aware adaptation of user interfaces with the application of machine learning algorithms.

While performing adaptation with machine learning one important trade-off must be considered, i.e. the benefits of performing such adaptation may overcome the costs that calculating it may have, e.g. if a significant performance impact is expected, the benefits for the usability must compensate such delay. This trade-off must be carefully analysed and discussed within each application scenario.

5.2 Summary

As potential algorithms for adaptation, we propose mainly: clustering, decision trees and regression.

Two scenarios were defined to illustrate possible applications: the transformation between task trees to AUI models, and the selection of widgets.

For the first scenario, we highlight a set of important tasks and defined some relevant criteria for their adaptation (i.e. grouping, selection, ordering): their workload, temporal demand, cognitive demand, physical demand and mental demand.

For the scenario of selection of widgets, we also presented a set of algorithms that have been proposed to support this task, and we highlight as important criteria to support the selection: the profile of the user (e.g. age, impairments, attention level), the characteristics of the screen (e.g.: dimensions, touch-based interaction, pen-based), the devices available (e.g. keyboard, mouse), and some characteristics of the environment in which the interaction occurs (e.g. stress level, stability, etc).

5.3 Future Work

As a future work we plan to analyse in depth the trade-offs posed by applying machine learning for adapting UIs. Mainly we would like to define some specific thresholds that indicate the association between costs and benefits in this scenario.

6 References

Hao-hua Chu , Henry Song, Candy Wong, Shoji Kurakake, Masaji Katagiri, Roam, a seamless application framework, The Journal of Systems and Software (2003)

K. Coninx, K. Luyten, C. Vandervelpen, J. Van den Bergh and B. Creemers, Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems, Human-Computer Interaction with Mobile Devices and Services ,Lecture Notes in Computer Science, 2003, Volume 2795/2003

Fowlkes, E. B., and C. L. Mallows.1983. A method for comparing two hierarchical clusterings. J. Am. Stat. Assoc. 78:553-569.

D. Fréard, E. Jamet, O. Le Bohec, G. Poulain and V. Botherel, Subjective Measurement of Workload Related to a Multimodal Interaction Task: NASA-TLX *vs.* Workload Profile, Human-Computer Interaction. HCI Intelligent Multimodal Interaction Environments, Lecture Notes in Computer Science, 2007, Volume 4552/2007, 60-69, DOI: 10.1007/978-3-540-73110-8_7

J. M. Gómez, "A Survey on Approaches to Adaptation on the Web," System, pp. 137–153, 2009.

González-Calleros, J.M., Guerrero-García, J., Vanderdonckt, J., Muñoz-Arteaga, J., Towards Canonical Task Types for User Interface Design, Proc. of Joint 4th Latin American Conference on Human-Computer Interaction-7th Latin American Web Congress LA-Web/CLIHC'2009 (Merida, November 9-11, 2009), E. Chavez, E. Furtado, A. Moran (Eds.), IEEE Computer Society Press, Los Alamitos, 2009, pp. 63-70.

Jacob Eisenstein and Angel Puerta. 2000. Adaptation in automated user-interface design. In Proceedings of the 5th international conference on Intelligent user interfaces (IUI '00). ACM, New York, NY, USA, 74-81. DOI=10.1145/325737.325787 http://doi.acm.org/10.1145/325737.325787

Q. Limbourg, J. Vanderdonckt, Transformational Development of User Interfaces with Graph Transformations, (2004) Proc. of the 5th International Conference on Computer-Aided Design of User Interfaces CADUI'2004, Madeira, Kluwer Academics Publishers, Dordrecht, 2004.

Luis Leiva. 2012. Interaction-based user interface redesign. In Proceedings of the 2012 ACM international conference on Intelligent User Interfaces (IUI '12). ACM, New York, NY, USA, 311-312. DOI=10.1145/2166966.2167028 http://doi.acm.org/10.1145/2166966.2167028

Francisco Javier Martínez-Ruiz. 2010. The triad-based design of rich user interfaces for internet applications. In Proceedings of the 2nd ACM SIGCHI symposium on Engineering interactive computing systems (EICS '10). ACM, New York, NY, USA, 345-348. DOI=10.1145/1822018.1822077 http://doi.acm.org/10.1145/1822018.1822077

McCracken & Aldrich taxonomy: McCracken, J. H. & Aldrich, T.B., 1984, Analysis of selected LHX mission functions: Implications for operator workload and system automation goals. (Technical Note ASI 479-024-84(b)), Fort Rucker, AL: Anacapa Sciences, Inc

Angel R. Puerta and David Maulsby. 1997. MOBI-D: a model-based development environment for user-centered design. In CHI '97 extended abstracts on Human factors in computing systems: looking to the future (CHI EA '97). ACM, New York, NY, USA, 4-5. DOI=10.1145/1120212.1120215 http://doi.acm.org/10.1145/1120212.1120215

S. Rubio, E. Díaz, J. Martín and J. M. Puente, Evaluation of Subjective Mental Workload: A Comparison of SWAT, NASA-TLX, and Workload Profile Methods, APPLIED PSYCHOLOGY: AN INTERNATIONAL REVIEW, 2004, 53 (1), 61–86

SEESCOA, Software Engineering for Embedded Systems Using a Component Oriented Approach, httap//www.cs.kuleuven.ac.be/cwis/research/distrinet/projects/SEESCOA/

Vanderdonckt, J. M. and Bodart, F. Encapsulating Knowledge for Intelligent Automatic Interaction Objects Selection, in Proc. of InterCHI'93. 1993: ACM Press.

Wickens, C.D. (1984). Processing resources in attention. In R. Parasuraman & D.R. Davis (Eds.), *Varieties of attention* (pp. 63–102). Orlando, FL: Academic Publishers.

Acknowledgements

- TELEFÓNICA INVESTIGACIÓN Y DESARROLLO, http://www.tid.es
- UNIVERSITE CATHOLIQUE DE LOUVAIN, http://www.uclouvain.be
- ISTI, http://giove.isti.cnr.it
- SAP AG, http://www.sap.com
- GEIE ERCIM, http://www.ercim.eu
- W4, http://w4global.com
- FUNDACION CTIC http://www.fundacionctic.org

Glossary

- http://www.serenoa-fp7.eu/glossary-of-terms/