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

Project no. FP7 – ICT – 258030

Deliverable 2.4.1 Criteria for the Evaluation of CAA of SFEs

SEVENTH FRAMEWORK PROGRAMME

Due date of deliverable: 31/08/2011

Actual submission to EC date: 31/08/2011

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

[PU]	[Pubic]	Yes

Document Information	
Lead Contractor	CNR-ISTI
Editor	Fabio Paternò, Carmen Santoro
Revision	v.5 (31/08/2011)
Reviewer 1	W4
Reviewer 2	
Approved by	
Project Officer	Jorge Gasós

Contributors	
Partner	**Contributors**
CNR-ISTI	G.Ghiani, F.Paternò, C.Santoro, C.Sisti, L.D.Spano

Changes			
Version	**Date**	**Author**	**Comments**
1	20/07/2011	Carmen Santoro, Fabio Paternò	Basic structure with Initial Comments…
2	10/08/2011	Carmen Santoro	First complete draft
3	26/08/2011	Fabio Paternò, Giuseppe Ghiani, Christian Sisti, Lucio Davide Spano	Extended draft with structure revised

4	29/08/2011	W4, TID	Internal review
5	30/08/2011	CNR	Final Version

Executive Summary

The main goal of this deliverable is to define criteria (both qualitative and quantitative) and potential benchmarks for evaluation of tools for Context Aware Adaptation (CAA) of service front ends (SFE) and the associated applications. These criteria are expected to provide useful indications for those who work in the field of context-aware adaptation of SFEs and become the backbone for systematic ongoing scientific research. The criteria will be applied first internally to the languages and tools developed in the project in order to conduct self-evaluation of the progress, but also externally with respect to competitors in order to provide a comparative analysis. Evaluation protocols and potential benchmarks will be made publicly available allowing other laboratories to compare results and advance the state-of-the-art.

The criteria will allow access to the proposed solutions both from the designers and the end-user viewpoints in terms of their effectiveness and satisfaction. We will consider usability of various project results, including the authoring tools and the adaptation at run-time. The usability of the model-based approaches will be investigated as well. The adaptation process will also be evaluated in terms of software engineering parameters (such as robustness, efficiency, portability, etc.).

Table of Contents

1 Introduction

1.1 Objectives and Overview

The main goal of this deliverable is to define criteria (both qualitative and quantitative) and potential benchmarks for evaluation in tools for Context Aware Adaptation of service front ends and the associated applications.

1.2 Audience

The audience for this deliverable is twofold: *the project team*, in order to provide concepts, methods, and tools that can continuously provide feedback on how well adaptation is supported; *the broader international developers and designers community*, that can be interested in applying such evaluation criteria to their context-sensitive tools and applications as well.

1.3 Related Documents

Deliverable D1.1.1 Requirements Analysis, provides some useful input for identifying the evaluation criteria.

Deliverable D1.2.1, on the architecture, provides useful indications about the project results that will have to be assessed.

A deliverable on the same topic is planned for M24 (D2.4.2) and will provide the opportunity to further refine this set of criteria based on the experiences done in the second year of the project.

1.4 Organization of this document

Section 1 (this introduction) presents the goals, audience, related documentation of this deliverable and the organisation. In Section 2 we provide an overview of what will have to be evaluated in the SERENOA project (design and run-time tools, languages for user interfaces, applications…). We then have dedicated sections focused on technical evaluation (Section 3), usability evaluation (Section 4), User Interface adaptation (Section 5). Section 6 then provides indications about how to structure a user test of adaptive systems and report, in section 7 interesting experiments in the scientific literature, which can be considered good best practise useful as reference point before concluding (Section 8)

2 What will be Evaluated

In SERENOA, we plan to evaluate the adopted solutions from two points of view: by using some relevant software quality factors (so doing a *technical evaluation*), and also by considering criteria aimed at evaluating user-oriented aspects (e.g. *usability and accessibility evaluation*).

We have identified a number of categories for the project results that will be subject to evaluation:

- The various software tools that will be developed within the project (e.g. tools for developing UIs, tools for specifying adaptation rules),
- Applications (the applications that will be generated using the above tools),
- Requirements (the requirements that have been identified by the project will be checked in order to evaluate to what extent they have been satisfied),
- Models and languages (UI models and related specification languages produced by the project will be evaluated as well),
- Architectural modules/components (specific architectural modules that will be subject to evaluation),
- Other possible solutions and algorithms (solutions/algorithms developed by the project and which can be evaluated "per-se", even without the existence of an underlying supporting tool).

In general, these results will be subject to both a technical evaluation and a usability evaluation. When we deal with a user-oriented evaluation, we have to assume that two main types of users will be considered: interactive application developers and end-users, depending on the type of system considered.

The tools that will be considered are:

- The authoring tools for developing multi-device interactive applications at design time (for which we plan to carry out both a technical and a usability evaluation for developers),
- Tools for specifying adaptation rules at design time (for which we plan to carry out both a technical and a usability evaluation for developers). In this category we can find both tools that adapt user interfaces within a given modality (e.g. from graphical desktop to graphical mobile devices) and tools that adapt to a different modality (e.g. from graphical Web user interface to vocal interface),
- Tools for customizing adaptation rules (which can be applied either at design time or run time by developers, but in some cases even by end users).

Regarding the applications, we can distinguish:

- Adapted applications produced by authoring tools.
- Adapted applications produced by runtime adapters.

In both cases the attention will be on an end-user evaluation in terms of usability and accessibility.

In the SERENOA D1.1.1, a number of functional and non-functional requirements for developers and end-users have been put forward. Such requirements will be considered in the evaluation as well. They provide some useful input for the evaluation criteria and can be further refined in the evaluation work, and in the end we will be able to indicate which of them are fulfilled.

Another important output of the project will be a set of model-based languages for multi-device interfaces and adaptation. This will also be an input for standardization in the W3C. Some partners provide useful input for this work, such as MARIA at CNR-ISTI, USIXML at UCL, MyMobileWeb at CTIC and LEONARDI at W4. Such languages can be assessed in technical terms and also in terms of usability for developers. For example in (Souchon, and Vanderdonckt, 2003), five criteria for assessment have been identified: abstraction level, amount of tags, expressivity of the language, openness of the language, coverage of concepts.

Other software components will be developed in the project, such as the context manager. Since this will be an infrastructure for supporting the overall adaptation environment its evaluation will be mainly technical, e.g. in terms of number of types of sensors and devices that can be supported.

Lastly, the algorithms for adaptation can be considered in the evaluation work as well. In this case the goal will be to compare different solutions for the same adaptation issues in terms of performance and coverage of the relevant cases.

3 Technical Evaluation

With technical evaluation, we plan to assess criteria like the features that are supported by the tools, tool performance, as well as the technological solutions and algorithms that have been identified in the project. In the scientific literature, a number of technical metrics coming from the software engineering field (see (ISO/IEC 9126), (van Vliet, 1999)) have been reported. In general, various aspects can be considered for each criterion:

- *Applied to*: for each metric, it is useful to clearly indicate to what the metric is expected to be applied. Possible values for this attribute could be: all the tools (if the metric can be applied to evaluate all the tools developed in the project), the name of a specific tool (if the considered metric is relevant only for some specific tool), specific architectural components, UI languages, etc.
- *Criterion name*: it is important to provide a meaningful name for the evaluation criterion.
- *Definition*: a definition of the metric, together with further explanations, if needed, for clarifying the associated concepts.
- *Measure*: what measure can provide a concrete, verifiable variable to be associated with the criterion. For instance, a way of quantifying interoperability could be the number of interoperable data formats that a tool can support.
- *Technique/Method of application:* possible technique(s) with which data associated with a certain criterion can be gathered.
- *Success/Acceptance*: it can be useful to determine whether a certain criterion goal has been achieved or not. It can be seen as a sort of threshold for deciding about whether the criterion has been met or not.

In the following we introduce a number of criteria and discuss them taking into account such aspects, when possible.

3.1 Interoperability

Interoperability is the capability of the software to interact (inter-operate) with one or more other specified systems. It provides a measure of the ability to use the results between the different tools.

Interoperability can be measured by data exchangeability, i.e.: the number of interoperable data formats used in the tool. Another possible measure is the number of platforms/tools (both developed in the project and relevant third-party tools) which the tool can interact with. In the project, we plan to identify and count the platforms which the SERENOA results can interact with.

3.2 Correctness

Correctness is the degree with which software performs its tasks as defined in the requirements specification.

For assessing this criterion, it is useful to calculate the Must-have and Should-have requirements that have been partially or totally fulfilled. It can be considered successfully verified if 100% of Must-have requirements have totally been fulfilled and if more than 50% of Should-have requirements have partially OR totally been fulfilled.

3.3 Adherence to standards

Adherence to standards is the ability of software to comply with standards, regulations, guidelines, conventions, etc. Thus, the measure of it requires counting the number of relevant standards, guidelines, etc., the system is compliant with. In SERENOA, we plan to consider in particular the W3C standards, such as the guidelines for accessibility, the definition of XHTML and HTML5, and also to create new standards for task model and abstract user interface model descriptions.

3.4 Portability

Portability is related to the capability of the software product to be transferred from one environment to another.

Thus, it requires an analysis of the number of different (HW/SW) environments the system supports, whether the software depends upon libraries unique to a particular HW/SW installation, how much effort would be required to transfer the program from one HW/SW environment to another. For example, in SERENOA the authoring environments should be able to run on Windows, Linux and Apple OS X and the adaptation to mobile devices should support at least the main mobile platforms (Apple, Android, Nokia).

3.5 Security

Security refers to the capability of the software product to protect information and data so that unauthorised persons or systems cannot read or modify them and that only authorised persons or systems are allowed to access them.

Security can be supported using various types of techniques. One is the capability to assign different privileges to different users for accessing the various objects and supporting input data validation, preventing SQL injection and cross-site scripting. It can be supported through techniques of data corruption prevention, although it would require to count the number of implemented functions capable of corrupting/destroying data and to check whether data corruption prevention strategies have been implemented accordingly.

3.6 Efficiency

The capability of the software product to provide appropriate performance relative to the amount of resources used under stated conditions and to guarantee a specified level of performance when used under specified conditions.

This means the degree with which software fulfils its goal without waste of resources/time (e.g. whether functions have been optimized for speed). We can distinguish: *Time behaviour*, capability of software to provide appropriate response and processing times when performing its functions under stated conditions and *Resource utilisation*, e.g. usage of memory for saving data.

3.7 Maintainability/ Changeability

Maintainability and changeability refer to the capability of the software product to be modified after a working version is delivered. Modifications may include corrections, improvements or software adaptation to changes in environment, in requirements and functional specifications. The modification to one module should have as less impact as possible on the others. It should be possible to modify the code without introducing unexpected errors.

This refers to ease with which changes can be made to satisfy new requirements or to correct deficiencies. Possible measures are:

- Mean Time To Change (MTTC), when an error is found, it measures how much time it takes to analyse the change, design the modification, implement and test it;
- Cost of change/total cost of system;
- Readability of source code (see Aggarwall et al., 2002);
- Documentation quality (see Aggarwall et al., 2002).

3.8 Extensibility/ Evolvability

Extensibility and evolvability refer to the ability to extend/change a system at minimum effort cost. For example, in an adaptive system it can be the ability to add new adaptation rules or modify old ones.

3.9 Modularity

Modularity refers to the logical partitioning of the software design that allows complex software to be manageable for the purpose of implementation and maintenance and that enable parallel work.

The degree of independence of the various components can be measured by Cohesion, i.e. by evaluating how well components of a module fit together or, in other words, the degree to which all elements directed towards the same task are contained in a single component. (Cohesion should be maximized). Another possible measure is Coupling, which indicates how much different modules have to communicate (Coupling should be minimized).

3.10 Reliability

Reliability refers to the capability of the software product to maintain a specified level of performance when used under specified conditions.

In this case, the measures can be: Mean Time Between Failure (MTBF), the frequency of software failure is measured by the average time between failures; Mean Time To Repair (MTTR), the criticality of software failure is measured by the average time required for repair (for example the time to restart an adaptation server) and Reliability ratio (MTBF/ MTTR).

3.11 Availability

Availability refers to how long a system is operational. This can be measured by the percentage of time a system is available, versus the time the system is needed to be available. Formally, the system availability can be defined as MTBF/(MTBF + MTTR).

3.12 Fault tolerance/Robustness

Fault tolerance and robustness refer to the capability of the software product to maintain the availability of the service even in case of errors or failures.

3.13 Scalability

Scalability refers to the ability of a system to handle growing amounts of work in a graceful manner or its ability to be enlarged to accommodate that growth. For example, the model based languages that we are going to adopt in SERENOA should be able to allow designers to specify even complex applications and not only small examples.

3.14 Testability

Testability refers to the capability of the software product to allow modified software to be validated. It can be supported through a set of built-in test functions.

3.15 Recoverability

Recoverability refers to the capability of the software product to re-establish a specified level of performance and recover the affected data in the case of a failure.

3.16 UI Languages Expressiveness

The expressive power of the language refers to the degree to which the languages can describe every aspect of the user interface (presentation, dynamic behaviour, content, …)..

3.17 Use of Model-Driven Approach

The extent to which the model-based approach has been exploited by the tools developed in the project.

4 Usability Evaluation

In this section, we provide a number of criteria aimed at evaluating more user-oriented aspects (e.g. usability, accessibility). It is worth noting that all such criteria (both technical and user-oriented metrics) are very often dependent on each other. Then, it may happen that when increasing the compliance with one criterion, the compliance of other criteria can be increased as well. For instance, for technical criteria, when increasing modularity maintainability will also likely be improved. Similarly, comprehensibility is strongly related to developer learnability. However, in other cases, this is not true: for instance, sometimes, in order to increase security, the resulting usability of the system decreases, or end user learnability is not directly impacted by software modularity. Therefore, taking into account such relationships will be important to reach a reasonable trade-off for such criteria, also taking into account the most relevant criteria for the project goals.

The ISO standard definition of usability is "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use”. In addition other aspects can have an impact on the overall usability of a system.

4.1 Effectiveness

Effectiveness refers to the accuracy and completeness with which certain users can achieve specified goals in particular environments.

This can be measured in terms of task success rate: for each task, the average percentage of users who successfully completed each task or in terms of percentage of users who were able to complete all the tasks. The actual task completion can be detected in various ways: by an external observer, by explicit indication of the user or through the automatic detection of the events associated with it. For every unsuccessful task, in order to increase the test reliability, the cause of the failure should be considered too. Errors caused by the user (e.g., wrong interaction) should be distinguished from application errors (bugs) and from the abandonment of the task (that takes place when the user refuses for some reason to complete it).

4.2 Efficiency

Efficiency refers to the resources consumed in relation to the accuracy and completeness of goals achieved.

The system should enable the users to perform functions with an acceptable amount of time, storage space, bandwidth, and other resources. The system should be efficient to use, which means that once the user has got familiar with it, a high level of productivity is possible. One possible measure is task completion time (or time-on-task): time (in seconds) between the start and the end of a task, measured for each task and user. This can be measured in various ways: a) Automatic data logging during user tests; b) Moderator/note taker uses a stopwatch to measure such times; c) Evaluator reviews a video recorded during the user testing. Another measure is the average number of errors made by each participant for each task. An error is an action useless for performing the current task. Also this measure can be detected by: a) Automatic data logging during user tests; b) Moderator/note taker observes the user during the test; c) Evaluator reviews a video recorded during the user testing. Other possible measures are time spent on errors and frequency of help use: number of times users used the help of the system, measured for each task and average.

4.3 Satisfaction

Satisfaction refers to the comfort and acceptability offered by the work system to its users and other people affected by its use.

This can be measured by the number of times the user expresses clear frustration or joy, which can be detected in several ways: a) Moderator annotates the information while observing the user during the test; b) Evaluator reviews the recorded video. Other measures can be given by after-test questionnaire (done immediately after usability task performance and at the end of a usability test session); Rating scale for satisfaction with functions and features, which can be done through: a) Satisfaction surveys or qualitative interviews to get user attitudes towards the software; b) System Usability Scale (SUS, a ten-item attitude Likert scale giving a global view of subjective assessments of usability.). Another relevant measure is the percentage of users that rate the product as better than a key competitor.

4.4 Learnability

Learnability refers to the capability of the software product to enable the user to easily learn its application.

It measures how much time and effort are required to become proficient with the software. It refers to the novices' ability to reach a reasonable level of performance rapidly.

It is possible to have Subjective measure (user feedback) in terms of a (often 5-point) Likert rating scale, which can be provided by an after-test questionnaire response. Another Subjective measure is the Number of learnability related user comments, which can be detected by a) Moderator annotates the information while observing the user during the test or b) Evaluator reviews the recorded video. There are also metrics that consider effectiveness of documentation and help, such as: Decrease in help commands used over certain time interval; what proportion of tasks can be executed correctly after using documentation/help. Other metrics are based on usability change: increase in efficiency over time, which require to assess and compare the change in usability over time (e.g. improvements in efficiency in different trials over time). A further measure is "How long does the user take to learn how to perform the specified task efficiently?" (time to achieve expert performance) or to compare the usability of a product for novice and expert users.

4.5 Memorability

Memorability refers to the ability for users to go back to the system and remember how to use it once they have been away from it for some time, without having to perform relearning. This requires measuring retention over time, which means that it should not take more than a certain amount of time depending on the system used to re-learn. Thus, it is required to test the system with the same users in different trials over time.

4.6 Comprehensibility

Comprehensibility refers to the capability of the software product to enable the user to understand a) whether the software is suitable, and b) how it can be used for particular tasks and conditions of use.

This can be measured through a Likert rating scale, which can be included in after-test questionnaires.

4.7 Error tolerance

Error tolerance refers to the ability of the software to provide users with relevant support in case of users' errors, in order to allow an easy recovery. An error is an action performed by the user that is not useful to the accomplishment of the current task.

Error tolerance can also be measured through a Likert rating scale, which can be included in after-test questionnaires.

4.8 Accessibility

Accessibility refers to the degree to which a product, device, service, or environment is available to as many people as possible, including persons with some level of impairment.

This can be measured through checking the capability of the software product to adhere to standards, conventions, style guides or regulations relating to accessibility (such as W3C Accessibility standards).

4.9 Attractiveness

Attractiveness refers to the capability of the software product to be attractive to the user. Also this aspect can be measured through a Likert rating scale, which can be included in after-test questionnaires.

4.10 Controllability/Programmability

Controllability and programmability refer to the degree of control the software provides to the user.

It deals with how users feel that they are in control of the software. It can be measured through a Likert rating scale provided by after-test questionnaire.

4.11 Flexibility

Flexibility refers to the multiplicity of ways the user and the system exchange information (Input/Output) in different forms without fixed task ordering.

4.12 Effective feedback

The system should offer informative and effective feedback about the effect of the interaction.

4.13 Tool support for usability evaluation

Usability evaluation is an important phase. For this purpose automatic tools are very useful to gather larger amount of usability data and support their analysis.

Remote evaluation (Hartson and others, 1996) implies that users and evaluators are separated in time and/or space. This is important in order to analyse users in their daily environments and decreases the costs of the evaluation without requiring the use of specific laboratories and asking the users to move. In particular, tools for remote Web usability evaluation should be sufficiently general so that they can be used to analyse user behaviour even when using various browsers or applications developed using different toolkits.

Ivory and Hearst (2001) provided a good discussion of tools for usability evaluation according to a taxonomy based on four dimensions: method class (the type of evaluation); method type (how the evaluation is conducted); automation type (the evaluation aspect that is automated); and effort level (the type of effort required to execute the method). CNR-ISTI is developing a tool, Web Usability Probe (Carta, Paternò, Santana, 2011), which allows the evaluator to create tests based on real Web sites instead of prototypes, giving the possibility to observe the user's interaction in a real environment. According to Ivory and Hearst classification, this solution is for usability testing, it captures logs generated client-side, supports automatic analysis and a number of visualizations to ease the identification of the usability issues, and only requires that users perform some predefined tasks specified by the evaluators. CNR-ISTI prefers logging on the client-side in order to be able to capture any user-generated events, which can provide useful hints regarding possible usability problems. Moreover, the tool allows the usability experts to analyse through some graphical representations of the logged data how users interacted with the user interface (UI). The tool works also for mobile applications, thus it can be useful to support usability evaluation of Web applications accessed through various types of interactive devices.

5 UI Adaptation-Oriented Criteria

Adaptation can be applied to various user interface aspects:

- Presentation, for the perceivable aspects (including media and interaction techniques choice, layout, graphical attributes, …)
- Dynamic behaviour, including navigation structure, dynamic activation and deactivation of interaction techniques, …
- Content, including texts, labels, images

Various combinations of adaptation strategies are possible, for example:

- Conserving (keep the arrangement/presentation of UI objects);
- Rearrangement (UI objects kept after adaptation but they are rearranged according to some techniques, e.g.: different layout;
- Increase (target device can provide more UI features);
- Reduction (less UI features);
- Simplification (UI objects kept, but with simplified representations, e.g.: images with lower resolutions);
- Magnification (opposite of simplification).

Thus, different adaptation tools can provide different results. Due to the importance of UI adaptation in our project, in this section we focus on criteria specifically relate to this aspect. To this end, we have identified a number of criteria that are relevant to adaptation. The discussion will follow the same description structure as before; for each identified criterion, we provide its definition, a suitable measure for assessing it, and the techniques that can be used for collecting the related data.

5.1 Predictability of UI adaptation

Users should be able to understand under which circumstances adaptation takes place, what UI parts are going to be adapted and where the user interaction results will appear after adaptation.

The goal is to be able to understand the result of the adaptation, which can be then assessed through a Likert rating scale, where respondents rate their level of agreement with respect to predictability.

5.2 User's Awareness of UI adaptation

This refers to the awareness of the user regarding the changes in the UI due to adaptation.

The goal is to be able to answer the following question: *“Is the user able to realise that a specific change in the UI was caused by adaptation?”* This can be also detected through a Likert rating scale, where respondents rate their level of agreement with respect to awareness.

5.3 Appropriateness of the adaptation

When the user realises that adaptation has been performed, does s/he judge the performed adaptation as appropriate?

The goal is to be able to answer the following question: *“Does the user think that the adaptation improves the quality of interaction with the system?”* This can be also detected through a Likert rating scale, where respondents rate their level of agreement with respect to appropriateness.

5.4 Timeliness of the adaptation

Timeliness of the adaptation refers to the application of adaptation in a timely manner (e.g., not too late) when there is an actual need to change some aspect of the user interface to better support the user.

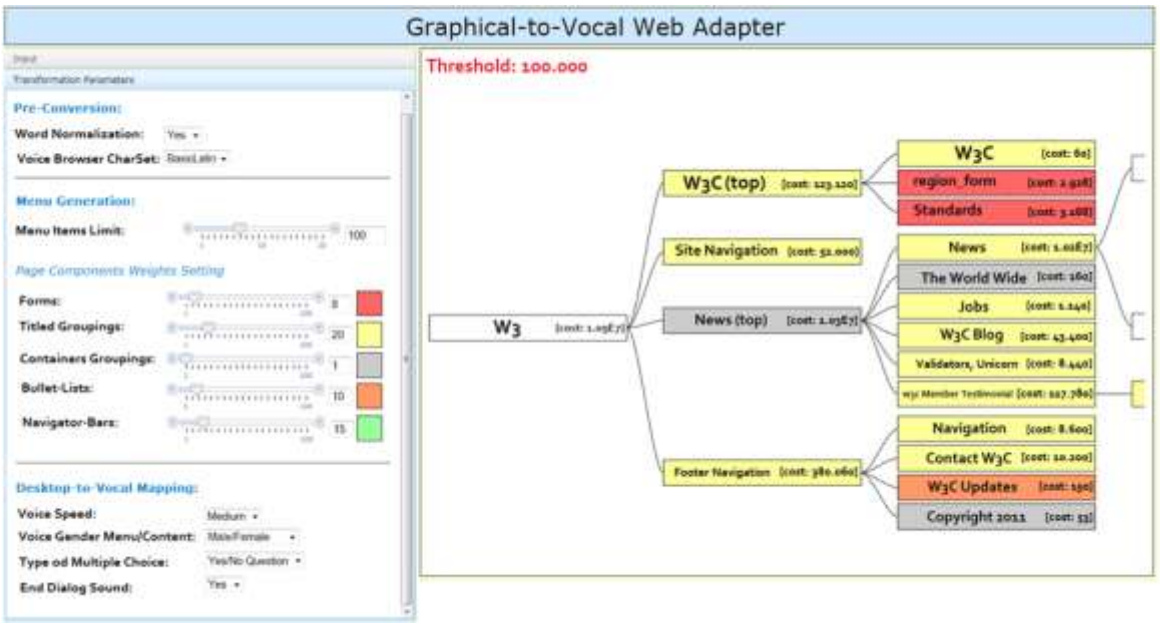
The goal is to be able to answer to the following question: *“Does the user think that when the adaptation occurs, then it actually improves the quality of the user interaction with the system?”* This can be also detected through a Likert rating scale, where respondents rate their level of agreement with respect to timeliness.

5.5 Controllability/Programmability of the adaptation

This criterion refers to the user's control over the adaptive process, namely the users’ ability to control both the circumstances that lead to triggering adaptation, and how adaptation is actually applied.

The capability of modifying the behaviour of the adaptation can be achieved through various techniques: by changing the values of parameters that determine changes in the adaptation results, or even by changing or adding the rules that determine adaptation.

A first example of an assessment of this criterion in the case of a tool supporting customization of desktop-to-vocal Web application adaptation (see Figure 1) carried out in the project is reported in (Paternò and Sisti, 2011).



Pre-Conversion:
Word Normalization: Yes
Voice Browser CharSet: BasicLatin

Menu Generation:
Menu Items Limit: 100

Page Components Weights Setting
Forms: 8 🟥
Titled Groupings: 20 🟨
Containers Groupings: 1 ⬜
Bullet-Lists: 10 🟧
Navigator-Bars: 15 🟩

Desktop-to-Vocal Mapping:
Voice Speed: Medium
Voice Gender Menu/Content: Male/Female
Type od Multiple Choice: Yes/No Question
End Dialog Sound: Yes

Threshold: 100.000

Graphical-to-Vocal Web Adapter
Input
Transformation Parameters
W3 [cost: 1.05E7]
W3C (top) [cost: 123.120]
Site Navigation [cost: 52.000]
News (top) [cost: 1.03E7]
Footer Navigation [cost: 380.060]
W3C [cost: 60]
region_form [cost: 2.918]
Standards [cost: 3.288]
News [cost: 1.01E7]
The World Wide [cost: 160]
Jobs [cost: 1.240]
W3C Blog [cost: 43.400]
Validators, Unicorn [cost: 8.440]
W3C Member Testimonial [cost: 117.780]
Navigation [cost: 8.600]
Contact W3C [cost: 10.200]
W3C Updates [cost: 150]
Copyright 2011 [cost: 53]

Figure 1: Customization of the Graphical-to-Vocal Web Adapter

5.6 Perceived Usefulness of the adaptation

This criterion refers to the user's perceived usefulness of the adapting system.

Usefulness can be dependent on the domain (e.g. learning rate for educational software, increase of sales for e-commerce systems, etc.).

5.7 Within-device consistency of UI adaptation

This criterion refers to the consistency of the UI design after adaptation with the design before adaptation.

In this case we consider that adaptation is still performed for the same type of device.

5.8 Across-device consistency of UI adaptation

This criterion refers to the level of consistency between the UI design after and before an adaptation following a device change. Device change implies that the user interface is migrated from one device to another.

An example is when desktop-to-mobile adaptation is performed. Some level of consistency on the mobile adapted version has to be maintained in order to avoid an excessive cognitive effort for understanding the new interaction model.

5.9 Continuity

This criterion refers to the possibility to easily continue the interaction after adaptation. Also in this case, we can distinguish the case when adaptation occurs in the same device or cases in which there is a change of device. Continuity can be considered at different levels: at *task* level, which means the ability to continue the task even after the user interface has changed due to adaptation; at *action* level, it refers to the possibility to continue, even after adaptation, the previously started physical action.

5.10 Adaptation Performance

This criterion refers to the time required to perform the adaptation. It is considered to have a strong impact on the user experience.

5.11 Adaptation Transition

This criterion refers to the behaviour of the user interface during the adaptation should be understandable and should allow users to realize what is happening.

6 Relationships between evaluation Criteria and Requirements/Scenarios

In this section, we introduce how we have preliminary considered the evaluation aspects with respect to the projects' identified requirements. We then highlight some relationships between the evaluation criteria identified in this document and the requirements/scenarios that have been identified in the SERENOA Deliverable D1.1.1.

The requirements identified in D1.1.1 provide a number of criteria according to which evaluation should be carried out so that we can reasonably state that the project has fulfilled its requirements. In the D1.1.1 document, the requirements have been distinguished between functional requirements and non-functional requirements for both *UI developers/designers* and *end users*. The latter distinction done on requirements (if they affect designers vs. end users) allow us to distinguish what can be subject to evaluation: on the one hand, (the UI designers/developer's point of view) we consider mainly UI tools, UI models, UI languages, architectural components, etc, evaluated both regarding their technical aspects and also usability aspects, while on the other hand (end-user), the main focus will be basically on user-oriented metrics.

Another point that we can highlight regarding the relationships between requirements and evaluation criteria is that different relationships can be identified between them, and also a certain amount of overlapping. Sometimes, there is a quite direct mapping between the criteria identified in this document and the requirements identified in D1.1.1. For instance, the "Learnability" requirement (which appears both for the developer and the consumer in D1.1.1) has a direct relationship with the "Learnability" criterion/metric identified in this document as part of usability evaluation. The same holds also for other requirements (see for instance portability, efficiency, accessibility, etc.). In other cases requirements identified in D1.1.1 cover aspects that have not been identified in this document as specific evaluation criteria (see for instance the requirements connected with the Agile methodology, reported in D1.1.1). In other cases the opposite is true: see for instance the whole number of different criteria connected with UI adaptation that have been identified in this document, and which do not always have a direct counterpart in the project's requirement list.

So, as a consequence of this preliminary analysis, it can be concluded that a combination of both requirements and evaluation metrics has to be considered, as well as a more thorough analysis of the requirements identified in the project in the view of evaluation (that can be planned for a next version of deliverable on requirements).

7 Analysis of Evaluation Experiments on UI Adaptation

Usability evaluation can be carried out following various approaches. In the project, we plan to apply some of them. In inspection-based evaluation one or more experts analyse the user interface in order to detect potential issues. In user-based evaluation users are directly involved in detecting the usability problems; often this is performed through user tests. Such tests need to be carefully designed and this is particularly important when considering UI adaptation. Thus, we have considered useful to analyse some examples of user tests in this area that have been reported in the scientific literature and that can be considered useful starting points for the user tests to plan in the project.

This analysis will provide useful insights on how to conduct such type of evaluations. However, before reporting such analysis, we judged it useful to recall some key points regarding empirical evaluation, which will be needed to understand the analysis itself.

7.1 Some basic concepts about empirical evaluation

When empirical evaluation of adaptive interactive systems is to be performed, controlled experiments are often considered. The general idea underlying them is that by changing one element (the *independent* variable) in a controlled environment (for example changes in the navigation styles or in the textual content), some effects will be produced (and measured) on the *dependent* variable (for example task performance time or number of errors).

However, in order to ensure that such studies produce significant results, it is important that they are correctly planned and designed. To this aim, a number of steps should be accomplished (see Gena, 2005):

1. **Identify and precisely state the research hypothesis that the study should verify** *(hypothesis testing)*. A (statistical) hypothesis is an assumption about a population aspect, which may be true or not. There are two main types of hypotheses: the *null hypothesis* (denoted by H0), the hypothesis that sample observations purely result from chance and (one or more) *alternative hypothesis* (H1), the hypothesis that observations are influenced by a non-random cause. Two types of errors can result from a hypothesis test: *Type I error*, occurs when the null hypothesis is rejected while it is true or *Type II error*, occurring when the researcher fails to reject a null hypothesis that is false. When planning the study, it is really important that the null and alternative hypotheses are precisely stated at the beginning of the study.
2. **Depending on the hypothesis, formulate a plan by identifying independent and dependent variables**.
   - The *independent variable (AKA factor)* represents the condition that is under the control of the experimenter (e.g. an interactive system having or not UI adaptation features), so it is independent of a subject's behaviour, yet determining a condition to which subjects undergo during the test.
   - The *dependent variable* is dependent on the subject's behaviour and should be measured consequently. Examples could be the time to complete a task, the percentage of task completed, etc.

   Ideally, only the independent variable should vary from condition to condition, though in reality other factors might vary along with the treatment differences. These confounding factors (AKA nuisance variables) might pose problems to the experiment results: if they come up to impact the behaviour under study, it will be difficult to distinguish what is determined by the manipulated variable and what is brought back by such factors. Regarding the ***type of data*** associated to variables, four main possible types can be considered:
   - ***Nominal*** data represent unordered categories of data, e.g. pink, blue, white, black.
   - ***Ordinal*** data describe orders (or *ranks* like 1st, 2nd, 3rd, etc.). It is important to note that with such data the intervals among the various measured items are not meaningful. An example is when the user has to select between poor, fair, good and excellent: since they are ordinal, only the relative order is important, so it is not possible to derive that e.g. the distance between two specific consecutive items is the same occurring between other consecutive values.
   - With ***interval*** data, not only the order between the values is relevant, but also the differences/magnitude between them. In addition, an interval scale does not include zero.

- ***Ratio*** data are just like the interval data, but there is absolute zero point. Examples are age, task completion time, etc. where zero indicates the absence of value.

As discussed later on, different types of data imply different kinds of statistical operations applicable. For instance, with nominal and ordinal data we can carry out χ^2 (Chi-square) tests and compute frequencies, whereas with interval and ratio data we can use descriptive statistics, t-test, ANOVA, etc.

3. **Select the participants**. It is very often impractical to test the design with the whole population of users. Then, a reasonable number of *representative* subjects of the population has to be identified (*sampling*). The idea behind is that by collecting data on such a sample, it will be possible to extrapolate/make inference about the larger population. One of the best manners to get a representative sample is *randomly* select the participants. When not possible, the sample could be selected using other criteria (e.g. *convenience*/*availability sampling*). Another critical choice is related to the sample size (which number of users can be enough, e.g. N>10).
4. **Conduct the experiment**. This mainly means collecting data using a particular experimental design. Subjects can be assigned to the various treatment conditions following different methods. An important decision is whether to compare in the study different data for each subject, or data from each subject to other subjects. So, basically the study could be between-subjects or within-subjects:
    - In the ***between-subjects*** design, an *experimental group* of subjects is assigned to the treatment, while another group of subjects, the *control group*, is assigned to a condition consisting of *absence* of the treatment. Therefore, in a between-subjects factor, each subject experiences only one level. Usually, this procedure is aimed at detecting differences between two experimental conditions (e.g. is the application designed with adaptive support more usable than the one without it?). One of the advantages of this type of design is that guarantees independence (i.e., there is no learning effects). However, it has a greater variability and requires more subjects.
    - The ***within-subjects*** design occurs when each subject experiences all the treatment condition levels. The advantages of such a method are that it requires fewer subjects and there is less variability of measures. However, one problem is the possibility of *carryover* effects (namely, the fact that participation in one condition might affect performance on another later condition). Two types of carryover effects can be mainly identified: *practice/learning* (the previous treatment positively affects the performance on a later one); *fatigue* (the previous treatment negatively affects performance on a later one). One way to mitigate such issues connected with ordering is using *counterbalancing* (e.g. subjects perform the assigned tasks in different orders).

    The most common experiment design is the *factorial design*, in which more than one *factor* (or independent variable) is considered. Factorial designs are described using "M x N" notation, in which M and N are the number of factor levels (factor level = specific value of independent variable) of the two independent variables (e.g., a 2 x 2 x 2 factorial design is a design with three independent variables, each one having two levels). A key point of this design is the joint manipulation of the two (or more than two) independent variables. If they influence each other, then an *interaction* is present. Also, there are other possible designs in-between the above two, e.g. a *mixed design* might contain both a between-subjects factor (e.g. gender) and a within-subject factor (e.g. repeated trials of tasks over time).
5. **Analyse sample data.**
    - *Use* ***descriptive statistics*** *to describe the data*. These statistics (e.g. mean, variance, standard deviation, etc.) are aimed at summarising a set of data. They can be divided into a) *measures of central tendency*: mean (the average value); median (the middle value); mode (the most frequent value); b) *measures of dispersion*: variance, standard deviation and standard error; c) *measures of association*: correlation (the extent to which two aspects are related one another). Another aspect to note is the *confidence interval*, namely the amount of error allowed in the data. Since statistics uses a sample for deriving predictions for the whole population, we expect a certain degree of error modelled through the confidence interval which gives a measure of the reliability of e.g. the sample mean as compared to the whole mean.
    - *Use* ***inferential statistics*** *to evaluate the statistical hypothesis*. In order to report significant results and make inferences, descriptive data might not be enough and some inferential statistics measures are required. Inferential statistics are designed to make inferences about larger populations and to assess the significance level of an experiment, which is the probability (*p*)

with which the experimenter is willing to reject the *null hypothesis* when it is actually correct. The accepted significance levels are 0.05 (p=5%, 'significant') and 0.01 (p=1%, 'very significant'). Choosing the right test to compare measurements is crucial, it must be done between two families of tests: *parametric* and *non-parametric*. Table 1 below summarises how it is possible to decide which test should be selected (in the table, K =samples of data to be compared).

- *Parametric statistics*. Many statistical tests are based upon the assumption that data are sampled from a Gaussian distribution: they are *parametric* tests. They generally consider interval or ratio measurements, continuous variables, and assume that data are normally distributed. Commonly used parametric tests are listed in related row of table 1. Among them we cite **ANOVA**, the analysis of variance, which is used to determine whether differences in dependent variables are due to the different independent variable treatments or due to chance factors. ANOVA can generate a significant value (the p-value) to indicate whether the difference between groups is significant, or just occurring by chance: in order to spot this, the statistical procedure called *F test* should be used. Another test that analyses the differences between two means is the ***t-test*** (while t-test is for comparing 2 samples, ANOVA is for more than 2 samples).
- *Non-parametric statistics*. They make no assumptions on the sample size and consider nominal and ordinal measurements and discreet variables. They are listed in the related row of the table 1 and include Wilcoxon, Mann-Whitney test, and Kruskal-Wallis tests. It is worth noting that conclusions drawn from non-parametric methods are not as powerful as the parametric ones. However, as they make fewer assumptions, they are more flexible, more robust, and applicable to non-quantitative data. They are generally used to investigate hypotheses about samples as a whole, rather than on some specific properties (e.g. means).
  The **chi-square test (χ^2)** is another test used to evaluate the significant values assumed by nominal data, to compare expected results with observed results.
- *Correlation*: when we want to analyse the relationships between two variables, if the value assumed by one variable is known, then the other variable might be predicted.
- *Post-hoc tests:* measurement conducted after the data have been examined and are used to limit errors. Among the most used post-hoc tests we cite e.g. Bonferroni correction.

6. **Interpret the results and draw the conclusions**. In this step, the experimenter has to interpret the results and then draw the opportune conclusions regarding the hypothesis testing.

7. **Prepare a report for presentation**. In this step, the experimenter should prepare a precise report of the experiment. Such a report should precisely describe the experiment so as to allow its repeatability. Among the literature considered we identified a number of key points that authors should at least cover in their presentation of the experiment, which are outlined in the following list:
   - Experiment Planning:
     - Goals of the experiment and its research questions.
     - Participant characteristics (e.g. procedure for recruiting and selecting the subjects).
     - Experimental Material (e.g. details of the material, such as questionnaires, and equipment used).
     - Selection of Tasks and Task list.
     - Hypotheses formulation, Variables, etc. (e.g. for each goal define the hypothesis - null and alternatives -, the dependent/independent variables-, etc.).
     - Experiment Design (e.g. between or within-subjects).
     - Procedure for conducting the study (e.g. what happened to the participants from when they arrived to when they left, the test settings/environment, the schedule and logistics of the experiment).
     - Data collection and analysis.
   - Result reporting
     - Descriptive Statistics.
     - Hypothesis Testing.
   - Discussion and Conclusions.

		Type of experiment						
		K=2, between subjects	***K>2, between subjects***	***K=2, within subjects***	***K>2, within subjects***	***Correlations***		
Type of data	***Numerical (interval & ratio)***	Student t-test	ANOVA	Paired t-test	ANOVA	Pearson	***Parametric tests***	**Type of test**
	Ordinal	Mann-Whitney	Kruskal-Wallis	Wilcoxon	Friedman	Spearman	***Non-parametric tests***	
	Nominal	Chi-square	Chi-square	McNemar test	Cochrane Q-test	Kendall		

Table 1: How to select a statistical test

In the following sections, we analyse some studies reported in the field of UI adaptation in order to understand how the evaluation of interactive applications having some form of UI adaptation has been reported in the scientific literature. This will offer us useful hints about how to plan such evaluations.

7.2 Evaluating Model-based Approaches

In the paper (Aquino et al., 2010), a MDE approach that generates multi-platform graphical user interfaces (e.g., desktop, web) has been subjected to an exploratory controlled experiment. The usability of UIs generated for the two mentioned platforms and used on multiple display devices (i.e., standard size, large, and small screens) was examined in terms of satisfaction, effectiveness and efficiency. An experiment with a factorial design for repeated measures was conducted for 31 participants, i.e., postgraduate students and professors selected by convenience sampling. The data were collected with the help of **questionnaires** and **forms** and were analysed using **parametric and non-parametric tests such as ANOVA with repeated measures and Friedman's test**, respectively. Efficiency was significantly better in large screens than in small ones as well as in the desktop platform rather than in the web platform, with a **confidence level of 95%.** Satisfaction tends to be better in standard size screens than in small ones. The results suggest that the tested MDE approach should incorporate enhancements in its multi-device/platform user interface generation process in order to improve its generated usability.

7.2.1 The Experiment

- **Goals** (Goals/Question/Metric method). The goal of the experiment was to *analyse* multi-device/platform graphical user interfaces generated by MDE *for* evaluating their usability *with respect to* satisfaction, effectiveness, and efficiency. The application was developed with OO-Method/OLIVANOVA. The user interfaces were derived from a SIU (Service Interaction Unit) and a PIU (Population Interaction Unit) were generated for two different platforms (e.g., C# running on .NET and JavaServerFaces running on Java) and used in three different devices (e.g., small, standard, and large screens).
- **Hypotheses**. A number of hypotheses were clearly stated in the paper. **Null hypothesis 1, H10**: when using interfaces automatically generated from PIUs, the user satisfaction is the same for different platforms and devices. **Alternative hypothesis 1, H11**: when using interfaces automatically generated from PIUs, the user satisfaction is not the same for different platforms and devices. **Null hypothesis 2, H20**: when using interfaces automatically generated from PIUs, the user effectiveness is the same for different platforms and devices. **Null hypothesis 3, H30**: when using interfaces automatically generated from PIUs, the user efficiency is the same for different platforms and devices. Authors were interested in knowing usability results of user interfaces generated from SIU, and usability results of user interfaces generated from PIU, each null hypothesis includes a condition about the used interaction unit. For reasons of space, authors listed just the null hypotheses related to PIUs.
- **Variables**. A number of **dependent** and **independent** variables were identified. **Dependent variables** considered were user satisfaction, effectiveness, and efficiency. In particular, *satisfaction* was measured with respect to the user's perceptions of *system usefulness, information quality, interface quality,* and

overall satisfaction (the latter one is an aggregation of the three other perceptions). All such measures were derived from answers to the Computer System Usability Questionnaire (CSUQ), expressed using a 7-point Likert scale in which 1 represents the best score (strongly agree) and 7 represents the worst perception score (strongly disagree). *Effectiveness* was measured by *task completion percentage*, which represents the percentage at which a task has been correctly carried out. *Efficiency* was measured by task completion percentage in relation to the time expended doing a task. The variables that were intentionally varied (**independent variables**) during the experimentation were (all on the nominal scale): *device* (small, standard, and large size screens); *platform* (C# running on .NET and JavaServerFaces running on Java); *interaction unit* (SIU and PIU).

7.2.2 The Experimental context

- **Subjects of study**. Subjects selected by convenience sampling, i.e., the nearest convenient persons were selected. Participation was voluntary and subjects did not receive incentives. 31 people participated in the experiment. Participation was anonymous. The subjects did not receive training. A demographic questionnaire was applied for characterizing subjects according to age, gender, study level, and experience with the various devices involved and with the use of applications generated with OLIVANOVA. All subjects used standard size screens previously to the experiment, 42% had experience using large screens, and 39% had experience using devices with small screens. Furthermore, 81% of the subjects had experience in the domain of the application used in the experiment, and 42% had experience using applications generated with OLIVANOVA. However, in this paper, the authors do not analyse if these demographic differences affect the perception of usability in the different devices and platforms.
- **Objects of study**. The objects studied were multi-device/platform graphical user interfaces generated by MDE. The experiment was conducted using an application that allows the expenses of the employees of an organization to be managed (Expenses Report). The application was developed using the OLIVANOVA technology. Two UIs of the application were selected for the experiment: one derived from SIU of the model of the application, the other one derived from PIU of the same model. The application was generated for two platforms: C# running on .NET (desktop platform) and JavaServerFaces running on Java (web platform). Therefore, there were four UIs to be evaluated. Also, there were three different set-ups of devices in which the UIs were evaluated: an ultra-mobile PC with a small touch screen and stylus; a PC connected to a standard size screen with mouse and keyboard; a PC connected to a large screen TV with mouse and keyboard. The characteristics of such devices (OS, screen resolution, etc.) were described in the paper.

7.2.3 The Experimental Design

The experiment was a **factorial 3x2x2** design with **repeated measures**. Each subject evaluated two user interfaces of the Expenses Report application, the one derived from SIU and the one derived from PIU. Each of these interfaces was evaluated in two platform versions (i.e., desktop and web) and each of these four combinations was evaluated in the three set-ups of devices (i.e., with a small, a standard, and a large size screen). The order in which subjects tested the different combinations was randomized. Twelve different tasks, 6 for PIU and 6 for SIU, were prepared so that subjects performed a different task for each of the 12 combinations of device, platform, and interaction unit. Although the 6 PIU tasks were different, they were similar regarding complexity. Also, the 6 SIU tasks were similar in complexity. The assignment of the task to perform in each combination of device, platform, and interaction unit was done randomly.

7.2.4 The Procedure

The study initiated with a short presentation in which general information and instructions were given. Then, a demographic questionnaire was applied. Afterwards, the subjects interacted with 12 user interfaces (3 devices x 2 platforms x 2 interaction units), carrying out a different task in each of them. For each task, the guideline presented some questions to the subjects: to answer these questions, the subjects had to interact with the application. The *task completion percentage* was derived from answers to these questions. The guideline also requested subjects to write down the time at which they started and completed each task, so *efficiency* was derived using these start and completion times. Furthermore, the guideline presented a Computer System Usability Questionnaire (CSUQ) to be filled out after completion of each task. Following the indications given by the designers of CSUQ the 19 numerical answers of the CSUQ were upgraded from the Likert scale to the interval scale. Then, the four perceptions of satisfaction (system usefulness, information quality, interface quality, and overall satisfaction) were obtained. Each subject spent approximately two hours to complete the experiment.

7.2.5 Comments

In SERENOA we plan to follow a model-based approach. This type of study provides useful indications for analysing the usability of user interfaces for various interaction platforms generated through the model-based authoring tools developed in the project. While in this study only graphical user interfaces were considered (in various screen sizes, small, standard, and large). In SERENOA we can consider also authoring applications using different modalities, such as voice.

7.3 Evaluating Adaptive User Interfaces

In the paper we consider in this section (Gajos et al, 2006) **4 types of UIs were evaluated: 3 adaptive UIs** *Split UI, Moving interface UI, Visual pop-up interface UI* (having three different levels of cost/benefit ratio: low cost/high benefit; moderate cost/high benefit; low-to-moderate cost/low benefit and also in the way the UI elements suggested by the adaptation support were promoted and highlighted within the UI) **+ 1 non-adaptive baseline UI. In particular, a toolbar user interface was considered in the experiment (see Figure 2).** In addition, 2 **Types of Adaptation models/algorithms** (namely, the manner in which the elements to promote in the UI are identified) were evaluated: *Recency-based (R) and Frequency-based (F)*. Two experiments were carried out and reported in the paper. The first one had the goal to elicit realistic subjective responses and then used realistic and complex tasks; the predictability of the adaptive algorithm was varied. The second experiment aimed at measuring the mechanical properties of the adaptation method: subjects were presented with a series of quick repetitive tasks and the accuracy of the adaptive algorithm was varied.

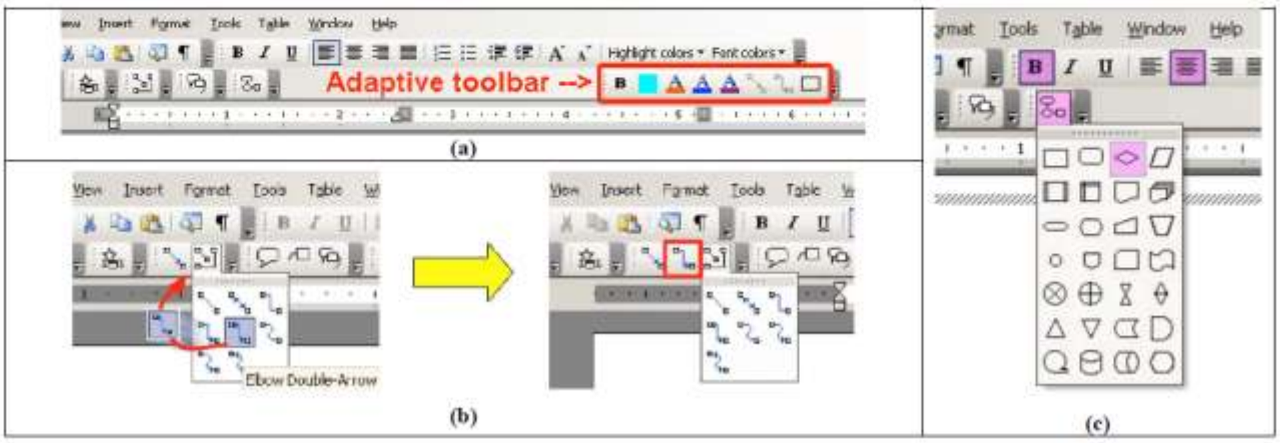


Figure 2: (a) The Split Interface; (b) The Moving Interface; (c) the Visual Pop-out Interface

7.3.1 1st Experiment

The first experiment was carried out by comparing the 3 Adaptive UIs with the baseline UI. It was basically a **between-subjects** experiment, and the **participants were** 26. The paper also reported the **subjects' characteristics** (in terms of age, gender, skills and what they received as compensation for their participation). Three **tasks** of medium complexity were chosen to mimic real world activities: Flowchart, Country and Poster. The **equipment** used was also described in the paper.

Regarding the **procedure and design of the experiment**, the users performed it in pairs. At the beginning of the session, a questionnaire was submitted to the users for collecting information, such as their computer experience and habits. An introduction to the study was also given to the participants. After that, instructions were provided to them, as well as a demonstration of the task. Users spent ten minutes in practicing some tasks with the non-adaptive UI. **The experiment was then a 4 (UI type evaluated) x2 (adaptation model) x3 (tasks)** having UI type and task as ***within-subject factors,*** and adaptation model as ***between-subject***

factor. Each of the 26 participants performed *4 isomorphic sets of 3 tasks*, each time with a different UI. **Counterbalancing** was applied across participants (using **Latin square design**) for presentation order. Between the sessions using different UIs, the experimenter explained how to use next UI. Since tasks were isomorphic and relatively distinct each other, interactions across conditions were judged more important than task ordering effects, so the order of task was kept constant for each condition. After each of the 4 task series with one UI, the participant had to fill in a **questionnaire** and after the last UI condition participants ranked the four UI types and explained their first and last choices. Finally, participants were debriefed. Each session lasted about 2 hours.

Regarding the **calculated measures,** three basic variables were first considered as possible **dependent variables:** satisfaction (evaluated through a **questionnaire**), overall preferences, and task times. However, task times were not sensitive enough, since authors did not find any statistical differences using this metric, so they excluded them from the analysis.

- Regarding **satisfaction, 4 (UI Type) x2 (Adaptation model) x11 (questions in the questionnaire) RM-ANOVA was used** *to analyse satisfaction questionnaire ratings* (UI type= within subjects; adaptation type=between-subjects). It resulted that a **significant main effect** for UI type F(3,69)=8.5, p<.001 and questionnaire item, F(10,230)=2.9, p=0.002 was found. In addition, **post-hoc tests** were used to understand when such differences appeared: they revealed that main effect for UI type was caused by significantly higher ratings for the Split Interface when compared to either no adaptation or the Visual Pop-out condition. However, there was no significant difference between Split and the Moving adaptive user interfaces: the Moving UI was not rated significantly higher than no adaptation. Final rankings of the 4 UIs, analysed with **Friedman non-parametric test**, highlighted a significant preference for the UI Split.
- An **estimation** of **participants' perceived cost and** benefits associated with the three UIs was computed, in particular a) average of responses to efficiency and performance questions (*measure of benefit*); b) mental demand, frustration and confusion (*measure of cost*). The results highlighted that the Split Interface was found most beneficial and least costly despite having lower theoretical benefit than the Moving Interface. As expected, the Visual Pop-out Interface was found to confer little benefit, but participants found it very distracting and assigned it a higher cost than the authors had expected.

Also, through their comments**,** participants confirmed that tasks did achieve the goal of high external validity, being realistic and engaging.

As a **summary of results,** this first study was judged important to observe participants interacting with the user interfaces on tasks that had high external validity. The high variability added by the cognitive decisions made within these tasks rendered task times insensitive to experimental manipulations (in fact, the task-time analysis did not show any significant differences across the 3 tasks). Since the satisfaction data showed **significant preferences** for the Split interface condition over no adaptation and highlights, authors assumed participants perceived benefits to this kind of adaptation that time measures for these tasks were not sensitive enough to capture. For this reason, authors decided to run a second experiment.

7.3.2 2nd Experiment

In this second experiment, the external validity requirements were relaxed by reducing the cognitive complexity of the tasks. Less realistic tasks (having the participant pressing more buttons while making fewer cognitive decisions) would allow to more carefully measure performance differences that might exist among the user interface types. Since in the first experiment authors found no differences between frequency and recency-based adaptation models, they decided to drop this variable from the study. Additionally, since there was a type of UI (Visual Pop-out), which was not possible to instrument to be as performing as the other two, they dropped that condition as well. Eight **Participants** were involved in this experiment.

Regarding the **Tasks and procedure**, at the beginning of each session, the experimenter explained and demonstrated the tasks and each user interface. Then participants worked on a practice task to familiarize themselves with the location of different commands and with the adaptive interfaces. Task in which participants were told which toolbar buttons they had actually to press were considered. In this task, the system presented an image of a particular command within the Word document. Each participant had to find and hit this button on the toolbar as quickly and accurately as they could. They then hit a “done” button, also presented within the document, and immediately got a new command to target. Each participant repeated this 52 times for each trial (but the first 12 were considered as a warm-up and were not included in the results). Two classes of tasks, which resulted in recency-based adaptive algorithm being either 30% or 70% accurate (i.e., correctly promoting the next button to be clicked by the user) in the case of the two adaptive conditions. Therefore, the experiment was a **3 (user interface type: no adaptation, Split Interface, or Moving) x 2 (accuracy: 30% or 70%) within-subjects design**. All conditions were fully **counterbalanced** to prevent the effects of training. The equipment used in the experiment was also described in the paper.

Regarding the **results**, task times, frequency of use and satisfaction were considered.

- For task times: **3 (user interface type) x 2 (accuracy: 30% or 70%) x 2 (trial order: 30% first or 70% first) RMANOVA, with user interface type and accuracy within-subjects, and trial order run between-subjects. Significant main effect** of user interface type, F(2,12)=8.545, p=.005. Pairwise comparisons using **Bonferroni correction** revealed that participants were significantly faster using the Split interface, p=.003, and marginally faster using the Moving interface, p=.073, than without adaptation. The two adaptive interfaces were not significantly different from each other. **Main effect of accuracy**, F(1,6)=8.859, p=.025. **Significant interaction between user interface type and accuracy**, F(2,12)=7.689, p=.007, driven by both adaptive interfaces resulting in faster performance (Split: p<.001, Moving: p<.001) with the 70% accuracy scenario faster than with 30%. **Significant interaction between accuracy and trial order**, F(1,6)=6.515 p=.043. Participants who saw the 70% condition followed by the 30% condition had a marginal decrease in performance, while participants that saw the 30% first showed vast improvements when they had the 70% condition (p<.001).
- Regarding **frequency of use**: further analyses of the Split UI usage data showed that the level of accuracy significantly affected the way participants interacted with the Split Interface, F(1,6)=10.361, p=.018.
- Regarding **satisfaction** and **user comments,** after each of the two sections of the experiment, the authors administered a brief **questionnaire**, asking the participants how easy each interface made it to find the functionality and how the participant felt it improved his or her efficiency. **A 3 (user interface type) x2 (# of questions) RM-ANOVA** was used to analyse the satisfaction questionnaire ratings. **The authors found a significant main effect** for UI type, F(2,30)=4.317, p=.023, explained via **post-hoc** analyses with **Bonferroni corrections** by the significant difference between split and unchanging (p=.035) and between moving and unchanging (p=.042). While participants felt that it was easier to locate functionality in the unchanging interface, they felt that both adaptive interfaces made them more efficient. In their post-experiment comments, participants focused primarily on three issues: ease of discovery, use of the adapted functionality, and the confusion caused by the adaptive interface. Many found the Split Interface not very useful because it required them to look in two distinct places for any piece of functionality (unlike in the first experiment, they saw no benefit in having frequently used functionalities grouped together). The Moving Interface was considered more convenient in that respect. The Moving Interface, however, caused items in pull-down menus to shift and also caused buttons on the toolbars to move horizontally as functionality was promoted or demoted. Even some of the participants who preferred the Moving Interface overall found this to be a concern.

7.3.3 Comments

This study provides a good example of how to perform usability evaluation of some adaptive features. Thus, it provides a good reference point for evaluating the adaptive applications that are going to be considered in SERENOA. The Deliverable 3.1.1 reports on many types of adaptive features that can be evaluated in a similar manner. They can vary in terms of granularity ranging from single UI elements (e.g. a label), to groups of elements (e.g. a form); to an entire presentation (e.g. by dynamically changing its layout). In such studies particular attention should be paid to understand for what types of tasks the adaptive support is suitable, whether its effectiveness depends on the type of user or device considered. More generally, we should consider for which context of use it is more appropriate.

8 Conclusions

8.1 Summary

This deliverable provides a first set of criteria useful to assess the results of the projects during their life cycle. Such results include authoring tools, run-time tools for adaptation and adaptive applications.

We have structured them in terms of technical and user-oriented evaluation criteria. We have also indicated a set of assessment criteria more specific to adaptation. We have also discussed how to address user tests to evaluate model-based development of multi-device user interfaces and adaptive user interfaces. We have referred to some papers that conducted relevant related studies in the case of our project, and which can provide useful indications to set-up the related user tests.

8.2 Future Work

In the previous sections we have identified a number of criteria that can be considered for evaluation in SERENOA. Our next step is to assign them a level of importance (low, medium, high) by taking into account the goals of the project. This will be useful to identify priorities about the criteria that can be considered the most important ones with respect to evaluation, and then better focus on them. In addition, we plan to perform a number of evaluations and report on them in the next evaluation deliverable (D2.4.2).

We also plan to identify a set of applications to use as benchmark for the adaptation techniques. Since Web applications are widely used we can consider the most frequently accessed Web applications for this purpose. In particular, one possibility is to consider pages belonging to the 100 most highly ranked well-known international web sites (according to http://www.alexa.com), such as w3.org, bbc.com, ebay.com, msn.com and yahoo.com.

9 References

Aggarwal, K.K.; Singh, Y.; Chhabra, J.K.; , "An integrated measure of software maintainability," *Reliability and Maintainability Symposium, 2002. Proceedings. Annual* , vol., no., pp.235-241, 2002

Aquino, Vanderdonckt, Condori-Fernández, Dieste, Pastor, Usability Evaluation of Multi-Device/Platform User Interfaces Generated by Model-Driven Engineering In Proc. of Empirical Software Engineering and Measurement ’10.

Bevan, N., Measuring usability as quality of use, Software Quality Journal, 4, 115-150 (1995)

T.Carta, F.Paternò, W.Santana, Support for Remote Usability Evaluation of Web Mobile Applications, ACM SIGDOC, Pisa, October 2011

Denis C. and Karsenty L. 2004. Inter-usability of multidevice systems – a conc eptual framework. In Seffah A. and Javahery H. (eds.) Multiple User Interfaces: Cross-Platform Applications and Context-Aware Interfaces. Wiley & Sons

Gajos, Czerwinski, Tan and Weld. Exploring the Design Space for Adaptive Graphical User Interfaces In proc. AVI 2006

Gena, C., Methods and techniques for the evaluation of user-adaptive systems, The Knowledge Engineering Review, 2005, Cambridge University Press. Doi:10.1017/S0269888905000299

Grossman, T., Fitzmaurice, G., & Attar, R. (2009). A survey of software learnability: Metrics, methodologies and guidelines. Paper presented at the CHI '09: Proceedings of the 27th International Conference on Human Factors in Computing Systems, Boston, MA, USA. 649-658.

H. R. Hartson, J. C. Castillo, J. T. Kelso, W. C. Neale: Remote Evaluation: The Network as an Extension of the Usability Laboratory. CHI 1996: 228-235.

ISO/IEC 9241-11 - Guidance on Usability.

ISO/IEC 9126 - Software Engineering - Product Quality

M. Y. Ivory, M. A. Hearst: The state of the art in automating usability evaluation of user interfaces. ACM Comput. Surv. 33(4): 470-516 (2001)

Jimenez Pazmino, P. and Lyons, L., An exploratory study of input modalities for mobile devices used with museum exhibits. Proceedings of the 2011 annual conference on Human factors in computing systems, CHI '11, pp. 895-904

Paramythis, A., Totter, A. & Stephanidis, C. (2001). A Modular Approach to the Evaluation of Adaptive User Interfaces. In Stephan Weibelzahl, David Chin, and Gehard Weber (Eds.) Empirical Evaluation of Adaptive Systems. Proceedings of workshop held at the Eighth International Conference on User Modeling in Sonthofen, Germany, July 13th, 2001, 9-24, Freiburg.

F.Paternò, C. Sisti, Model-Based Customizable Adaptation of Web Applications for Vocal Browsing, ACM SIGDOC, Pisa, October 2011

Rubin, J., Chisnell, D. Handbook of Usability Testing: How to Plan, Design, and Conduct Effective

Tests, Wiley, 2008.

Seffah, A., Donyaee, M., Kline, R.B., Padda. H.K.,Usability measurement and metrics: A consolidated model. Software Quality Journal (2006) 14: 159–178.

Souchon, C and Vanderdonckt, J, 2003. A Review of XML-compliant User Interface Description Languages. In the *Proceedings of 10th International Conference on Design, Specification, and Verification of Interactive Systems (DSV-IS 2003)*, 377-391. http://www.isys.ucl.ac.be/bchi/publications/2003/Souchon-DSVIS2003.pdf

Tullis, T., and Albert, B. Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics, Morgan Kaufmann, 2008.

van Vliet, H. "Software Engineering: Principles and Practice (2nd Edition)" Wiley, 1999.

Acknowledgements

- TELEFÓNICA INVESTIGACIÓN Y DESARROLLO, http://www.tid.es
- UNIVERSITE CATHOLIQUE DE LOUVAIN, http://www.uclouvain.be
- CNR-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