Improving Access to Services through Intelligent Contents Adaptation: The SAPI Framework

Nicola Capuano[1,2], Gaetano Rocco De Maio[3], Pierluigi Ritrovato[1,2], and Giacomo Scibelli[4]

[1] Research Center on Software Technologies, University of Sannio, Italy
[2] Dept. of Information Engineering and Applied Mathematics, University of Salerno, Italy
`{capuano,ritrovato}@crmpa.unisa.it`
[3] Information Technology Services S.p.a., Italy
`gaetano.demaio@its.na.it`
[4] Poste Italiane S.p.A., Italy
`Scibel19@posteitaliane.it`

Abstract. Nowadays, every public or private company has to provide the access to their services through Internet. Unfortunately, the access channels and devices increase both in numbers and heterogeneity. We started few years ago with a PC, wired connected to Internet, moved to wireless access through mobile phone and looking, in the next future, at wearable devices. If we also would like to take into account the user's preferences and his possible handicap we easily realise that combinations increase exponentially making impossible to adapt everything. The paper presents a rule based framework allowing to automatically adapt contents and services according to device capability, communication channel, user preferences and access context.

Keywords: Hypermedia Adaptive Systems; Context aware adaptive systems; Rule based adaptation systems.

1 Introduction to SAPI Adaptation Framework

In order to satisfy the growing request by the most people concerning with improvement and access facilitation to contents and services provided by the PosteItaliane S.p.A[1] the SAPI (Sistema Automatico Per Ipovedenti – Partially Sighted Automatic System) Project was proposed. SAPI aims to develop a framework for providing blind and half-blind users with intelligent services and contents, turning one's attention to adapting services and interfaces to *Situation*. In SAPI Situation is intended for the user-context and includes users' needs and features, environment, device and network.

SAPI belongs to the family of MultiChannel and MultiModal Adaptive Information System context-aware, so it has also implications for accessibility. SAPI pays great attention to the user interface design [1] [2] in order to satisfy advanced

[1] Poste Italiane is biggest mail service provider in Italy with more than 14.000 postal office, 200 sorting hubs, about 5000 ATM and about 50.000 POS on the country.

accessibility and adaptation requirements. In particular SAPI adapts its user interfaces in *batch* and at *runtime* (*on-the-fly*). In batch it uses evolution algorithms, while runtime, every time user-context changes it adapts executing the right adaptation rule. Since in SAPI a user interface is a collection of *Entities*, adaptation rules act on them and since entities are provided with semantic description, adaptation rules are described through an abstract and generic rule description language.

The flexibility and the efficiency of the system have been tested over practical scenarios of use. In case of need, the rules will be modified without modifying the application which uses them. Moreover using a rule language semantically interoperable and expressive allows an easy use and management of rules. So it will be possible to deal with the rule discovery using an innovative approach and, within the composition of adaptation rules, it will be possible to solve many problems about rule conflicts and find the best adaptation path. In fact, in SAPI, the description of a rule allows also to build adaptation paths in order to satisfy a more complex adaptation goal. The remainder of the paper is organized as follows. In section 2 there is a briefly description of the adaptation approach in SAPI. In section 3 we give some definitions and then illustrate the adaptation model which allows to define the new rule description language. Section 4 analyse the state of the art comparing current approaches with SAPI ones. Finally, in section 5 we describe how the adaptation engine works and illustrate some results.

2 Presentation Adaptation in SAPI

SAPI provides its users with many services each of them has a business logic, a workflow of activities, and a presentation logic that presents the results coming from the business logic in a specific user interface. Each service is able to adapt itself adapting its business and presentation logic. While the business logic is only adapted in batch, presentation logic is both adapted in batch and on the fly. Since the presentation logic is the composition of several user interfaces which, as indicated above, are collections of entities, adapting presentation logic means adapting entities. The entity is the atomic elements of SAPI and constitutes the basic element to compose a service. The Entity is a digital object (button, inputbox, hyperlink, etc.) or a media resource (text, audio, video, image, etc.), provided with a semantic description. It identifies a content which can have different versions and features but the same semantics. For each Entity there are two semantic sections: the first, called declarative section, contains metadata and relations between entities used for describing the content and related use; the second, called rule section, contains information about rules that allows an external actor to adapt on the fly the content to the specific situation. The semantic description is written in OWL.

Figure 1 shows the model of an Entity. An entity can be atomic or composed. A composed entity groups many atomic or composed entities and has: a navigation model that indicates what are the previous and next entity; an interaction model that indicates the logic order in which the entity can be presented to the users; a presentation model that indicates the layout of the presentation. Finally every entity has one or more adaptation rules that can be applied on it in order to satisfy user needs. Every rule is implemented and stored in the rule repository.

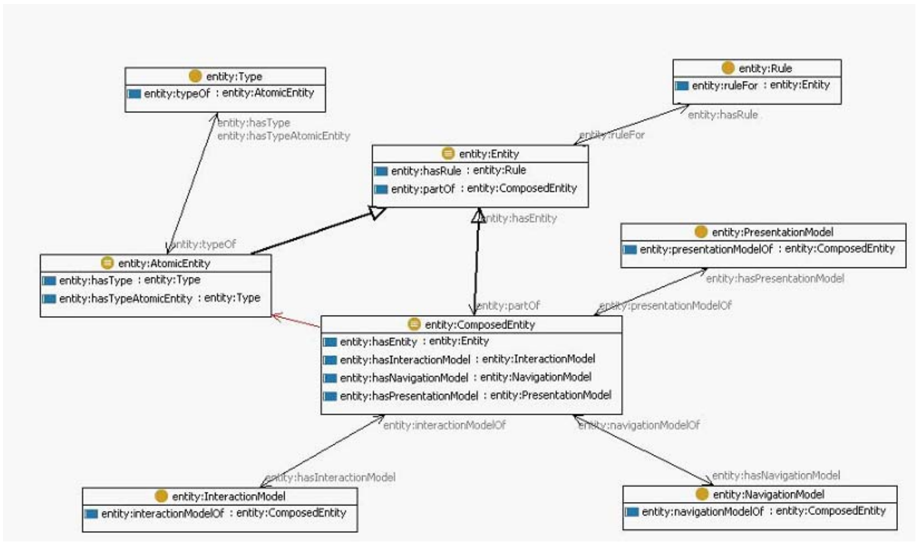

entity:Type
entity:typeOf : entity:AtomicEntity
entity:Rule
entity:ruleFor : entity:Entity
entity:hasType
entity:hasTypeAtomicEntity
entity:hasRule
entity:ruleFor
entity:Entity
entity:hasRule : entity:Rule
entity:partOf : entity:ComposedEntity
entity:hasEntity
entity:typeOf
entity:PresentationModel
entity:presentationModelOf : entity:ComposedEntity
entity:AtomicEntity
entity:hasType : entity:Type
entity:hasTypeAtomicEntity : entity:Type
entity:hasPresentationModel
entity:partOf
entity:presentationModelOf
entity:ComposedEntity
entity:hasEntity : entity:Entity
entity:hasInteractionModel : entity:InteractionModel
entity:hasNavigationModel : entity:NavigationModel
entity:hasPresentationModel : entity:PresentationModel
entity:interactionModelOf
entity:navigationModelOf
entity:hasInteractionModel
entity:hasNavigationModel
entity:InteractionModel
entity:interactionModelOf : entity:ComposedEntity
entity:NavigationModel
entity:navigationModelOf : entity:ComposedEntity

Fig. 1. Entity Model

Figure 2 shows the main SAPI's framework components involved in the adaptation. Every entity is stored in the "Entity Repository" which is accessible from the "Service Provider" (SP), a module responsible for the management and execution of the business and presentation logic. Execution of business logic consists in executing one o more activities, each of them refers at least one or more entities as results or as container to put results.

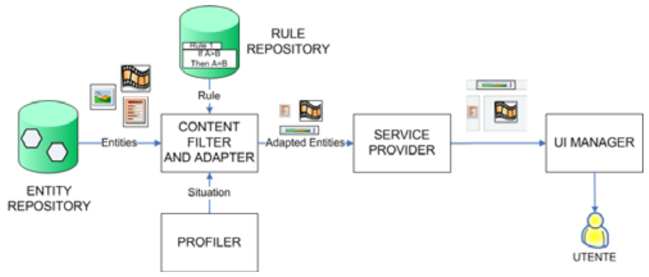


Fig. 2. Adaptation Architecture of SAPI

Once the entity is selected and set, before it could reach the SP have to be sent to the "Content Filterer and Adapter" (CFA) to adapt it to the specific situation. CFA is a rule engine that fires the right rule in order to adapt the entity to the situation that is sent from the "Profiler" which gets information about the user, the environment, the network and the device, querying an ontology and using physical sensors.

When Content and Adapter Filterer receives the entity, it analyses entity's description, decides which of its rules must be fired and selects them from the rule repository. Finally the entity will be sent to the UI manager which constructs the interface and presents it to the user.

3 The Adaptation Model

Before defining the adaptation model is necessary to fix some concepts. In particular, the concept of profile and stereotype, which contains the information the CFA needs to adapt entities, and the concept of adaptation and adaptation rule. Let's start with some definitions.

3.1 Core Definition

Definition – Profile: Given C = $\{c_{1,\dots,}c_n\}$, a set of characteristic, and Dc_i, the domain of values which i-th characteristic can assume; an instance of profile P is every n-tupla î belonging to the set P = Dc_1 X ...X Dc_n.

In SAPI we have defined the following characteristics:

- C_U are the characteristics of user: preferences, behaviours, abilities, etc.;
- C_E are the characteristics of environment: brightness, noise, etc.;
- C_D are the characteristics of device: CPU, OS, screen size, color depth, etc.;
- C_N are the characteristics of network channel: delay, bandwidth, etc..

Definition – Stereotype of a Profile: The stereotype of a profile P = Dc_1 X ... X Dc_n is a profile S = Sc_1 X ... X Sc_n which satisfies the next conditions:

- has the same characteristics of P;
- Sc_i is a subset of $Dc_{i,}$ i = 1,..., n;
- is consistent;
- identifies a class of instances of profile very similar among themselves;
- is dissimilar from other stereotypes.

Definition – Adaptation: The adaptation of an entity E in the state S_0 of a given Situation S is every sequence of conditioned actions such as, every time S changes, is able to lead E in a state compatible with S in a deterministic way. Because it will be useful in a short time, once given the definition of adaptation, is possible to give the definition of event too.

Definition – Event: An event is every change of state of the situation S represented by an instance of profile.

Definition – Adaptation rule: An adaptation rule is one of the conditioned action executed in order to adapt one or more entities. According to the literature on Active Database in SAPI the adaptation rule follows the Event Condition Action (ECA) formalism.

Definition – Adaptation model: An adaptation model M_a is a set of adaptation rules R_a = $\{r_1, \dots, r_n\}$ whose *termination* and confluence are guaranteed.

Since it is an unsolvable problem, Termination is often guaranteed defining a sufficient number of constraints on the properties of the rules which constitute the adaptation model [3].

3.2 The SAPI Adaptation Rule Model

In order to specify an abstract and generic model to describe adaptation rule, we started from results of two research areas: *Adaptive Hypermedia Systems* (AHS) and *Active Database*. Brusilovsky in [11] classified adaptation methods and techniques basing on main concepts which mark an adaptation rule: *Adaptation Goals* (**why**); *Something* (**what**); *Situation* (**to what**); *Methods and Techniques* (**how**).

Starting from these concepts, it is possible to define an abstract model which allows to describe a generic adaptation rule, i.e. a rule on which basis it is possible to adapt *something* (of the *Universe*) on a given *situation* (of the *Universe*) and/or on changes of a given situation which the rule is sensitive (*situation-awareness*). In SAPI *something* refers to the Entity.

Before starting to work out the model it is right to do a distinction between absolute transformation ability and relative transformation ability of an Entity.

Absolute transformation ability: the absolute transformation ability of an Entity is the set of Entity transformation methods which are situation-independent. These methods ignore the *situation* concept, so they are application-independent .

Relative transformation ability: the relative transformation ability of an Entity is the set of Entity transformation methods which are able to generate a new version of the Entity as congenial as possible to a given situation. Differently from the previous case, these methods must have consciousness about the concept of situation and its coding. So they are application-dependent. Figure 3 shows an example of relative transformation ability.

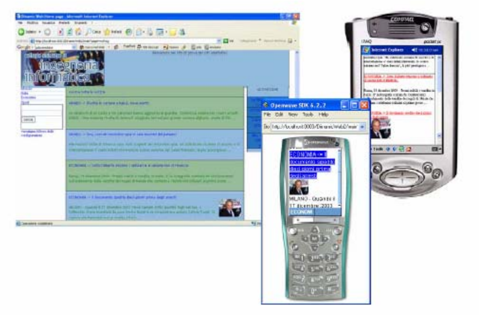


Fig. 3. Example of relative adaptation ability to three different situations

For the *activation* of an adaptation rule we assume that an adaptation rule can be fired, if and only if, some *conditions* (*Pre-Conditions*) are satisfied, in the following cases (*event/triggers*): on explicit request made by an actor (*Adaptor*); whenever an Event Detector finds a *situation* change (*sensor*); as a *propagation* of a rule which is able to trigger an internal event.

As a direct consequence of above, a formal description of adaptation rules comes down naturally. We show this description through the UML class diagram represented in Figure 4. For the sake of simplicity, all of possible triggers of an adaptation rule are considered sub-classes of the EVENT class.

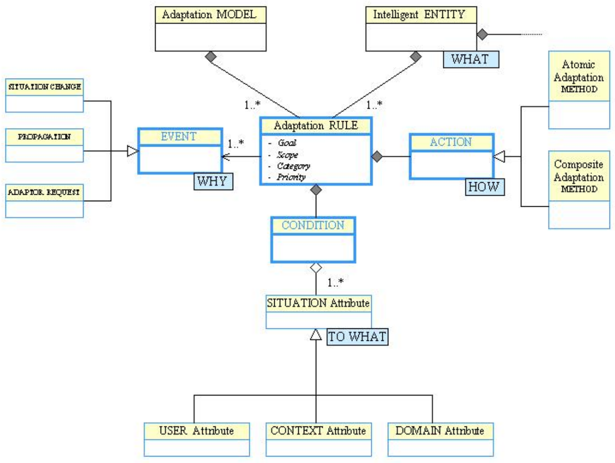


Fig. 4. SAPI Adaptation Model

Definition – Horn Clause: In mathematical logic, a Horn clause is a clause (a disjunction of literals) with at most one positive literal.

A Horn clause is a rule if and only if it has one, and only one, positive literal and at least a negative literal so that: $\sim A_1 \vee \sim A_2 \ldots \vee \sim A_k \vee C$ which is equivalent to $(A_1 \wedge A_2 \ldots \wedge A_k) \Rightarrow C$, i.e. an implication as *if Antecedent then Consequent* with one or more conditions and just only one conclusion.

So an *Adaptation Rule* Ar of one *Entity* En on a given *Situation* S, fireable when the event Ev occurs, in which the *implication* is an *Horn Rule*, is defined as follows:

$$\text{Ar: on Ev, Cd} \Rightarrow \text{Ad (En, S)} \qquad (1)$$

where:

- Cd is a condition consisting of a union of atoms as $C_i \gamma C_j$ or $C_i \gamma K$ where:
  - K= costant,
  - γ = comparative operator
  - C_i and C_j characteristic of S;
- Ad (En, S) = adaptation of En to situation S.

Therefore the intuitive meaning of an adaptation rule is: when an event Ev occurs on a given situation S, if the condition Cd is verified then do the adaptation of the Entity En on situation S. Basing on the (1) it is possible to observe that an adaptation rule is a specific ECA Rule (reaction rule) which paradigm, in his most generic form, can be considered as the next:

On Event *if* Condition *then do* Action(s)

For example:

On *Event: = < Brightness_ Sensor_Value <= Low_Threshold >*

If *($(C_D = (d_1, d_2, \ldots, _{PDA}, \ldots, d_k))$* ***AND*** *$(C_E = (e_1, e_2, \ldots, _{NIGHT, MOBILE}, \ldots, e_l))$* ***AND*** *$(C_N = (e_1, e_2, \ldots, e_l))$* ***AND*** *$(C_U = (u_1, u_2, \ldots, _{AGE > 60}, \ldots, u_n))$)*
Then *(Adaptation Method = A_{FE})*

Where A_{FE} uses *font enhancement* as adaptation technique. This adaptation rule says that when a man at least 60 year old uses a PDA at night the system must increment the font dimension so it will be easier readable.

4 Related Work

In the following we shortly present the state of the art on two relevant research topics: Adaptive Systems and rule descriptions.

4.1 Adaptive Systems

Adaptation of content and user interfaces in order to achieve accessibility is a great challenge that the research community have been studying for year, although the laws on accessibility are very recent. In [8] a tool for the automatic generation of adaptive Web sites is presented. FAWIS relies on two basic notions: the profile, used to model a variety of contexts and their characteristics, and the configuration which describes how to build the various levels of a Web application (content, navigation and presentation). These are respectively similar to SAPI profile and SAPI entity, which is richer thanks to its semantic description. The automatic adaptation of content delivery is achieved by means of an original notion of production rule that allows to specify declaratively how to build a configuration satisfying the adaptation requirements for a given profile. The module responsible of this work is the Context Manager. It selects the adaptation rules from a repository and, avoiding conflicting among them, generates the configuration that describes the final adaptation. This is a solution a little bit different from SAPI. SAPI repository of adaptation rules contains the implementation of rules but not the description that is part of the entity.

[9][10] propose an ontology-based approach to adaptation. The work in [10] defines a user model (an XML file) that consists of concepts and attributes and uses adaptation rules in order to perform updates of this model. The adaptation rules are production rules and define how the user model is updated. For example, when the user accesses a page (or an object included in a page) the rules associated with the access attribute are triggered. The approach proposed in [9] tracks user interactions and tries to present the content chosen by the user. This work is done taking in account the relationship existing among this content that are represented by entity with a semantic description. Since the adaptation is carried out using SWRL, [9] uses production rules too. SAPI uses the approach proposed in [10] only to guarantee the evolution of the user profile and implements the approach proposed in [9] through the interaction model of each entity. In addition SAPI doesn't use production rules but ECA rules and instead of presenting every time a different content, changes its presentation (for example summarizing it or changing to another version) without changing the entity and its description.

4.2 Rule Description Languages

Rules are generally fragment of knowledge *self-contained* which are usually used in a form of reasoning. They can specify, for example: static or dynamic integrity constraints of a system (*integrity rules*); derivations of new concepts (*derivation rules*); a reactive behaviour of system in reply to particular events (*reaction rules*).

Furthermore, rules can be described with three abstract level:

1. *Computation Independent Business Domain Level* (CIM in MDA of OMG) – rules are described in a declarative way using an informal or visual language;
2. *Platform Independent Operational Design Level* (PIM in MDA of OMG) – rules are described using a formal language or a computational paradigm which is easy translatable in instruction that a software system can execute.
3. *Platform Specific Implementation Level* (PSM in MDA of OMG) – rules are described using a language of a specific execution environment such as Oracle 11g, Jess 7.1, JBoss Rules (aka Drools), XSB 3.1 Prolog, o Microsoft Outlook Rule Wizard.

In PIM can be included *Rule Markup Languages* which main goal is to allow publication, deploying, reuse and rule interchange on the Web and distributed system. Rule markup languages also allow to describe rule in a declarative way and as modular unit self-contained and interchangeable between different systems and tools.
Among the main rule languages, actually available or developing, we mention RuleML[4], SWRL[5], R2ML[6].

RuleML Initiative started in 2000 to define a markup language able to support different kind of rules and different semantics. Even though it is an XML-based language and allows to describe all kind of rules, the latest version RuleML 0.91 (2006) hasn't yet a syntax for integrity rule and reaction rule. In order to get around this was proposed Reaction RuleML[7] which defines new constructs within separated modules which are added to the RuleML family as additional layers.

Semantic Web Rule Language (SWRL) is based on a combination of the OWL-DL and OWL-Lite sublanguages of the OWL Web Ontology Language with the Unary/Binary Datalog RuleML sublanguages of the Rule Markup Language. Its rules are of the form of an implication between an antecedent (body) and consequent (head): whenever the conditions specified in the antecedent hold, then the conditions specified in the consequent must also hold. Rules must be safe, i.e. variables that occur in the antecedent of a rule may occur in the consequent.

The REWERSE II Rule Markup Language (R2ML) allows interchanging rules between different systems and tools. It integrates the Object Constraint Language (OCL), a standard used in information systems engineering and software engineering, SWRL and RuleML and includes four rule categories: derivation rules, production rules, integrity rules and ECA/reaction rules.

Even though these languages allow to describe rules, to execute them is necessary an execution environment. For the experimentation in SAPI, we used JBoss Rules (aka DROOLS), an open source framework that provides standards-based business rules engine and business rules management system (BRMS) for easy business policy access, change, and management. It allows to encode business rules in the IT application infrastructure and supports a variety of language and decision table inputs, making it easy to quickly modify the business policies. It also simplify applications by separating business policy or rules logic from process, infrastructure, and presentation logic. This modularity enables to develop, deploy, modify and manage a business process' rules without modifying code nor rebuilding the application. Using JBoss Rules it was possible to translate the rule adaptation model in a set of executable rules. The code fragment below shows the code used to translate the previous adaptation rule which says that when a man at least 60 year old uses a PDA at night the system must increment the font dimension so it will be easier readable.

```
package sapi.adapter.rules
import sapi.adapter.ws.EntityAdapter;
import sapi.KnowledgeManager.bean.Entity;
import sapi.KnowledgeManager.bean.DeviceContext;
import sapi.KnowledgeManager.bean.EnvironmentContext;
import sapi.KnowledgeManager.bean.NetworkContext;
import sapi.KnowledgeManager.bean.UserContext;
rule "font_enhancement"
    salience 10
    no-loop true
    agenda-group "content_style"
    dialect "java"
    when
      $ent: Entity();
      $DC: DeviceContext();
      $EC: EnvironmentContext();
      $NC: NetworkContext();
      $UC: UserContext();
      (DeviceContext(device == "LowBrightness") and
      EnvironmentContext(environment ==
  "LowBrightness") and
       EnvironmentContext(ubiquity == true) and
       UserContext (age >= 60))
then
      $ent = EntityAdapter.fontEnhancement($ent);
end
```

This rule is stored in a file and if someone want to change it, it is not necessary to change the application or re-compile it.

5 Adaptation Examples

In the following we present some adaptation experiments carried out using the SAPI framework prototype.Keeping in mind the components presented in Figure 2, SAPI's components communication is based on web service, thus when SP needs contents, it invokes the CFA web Service which implements the adaptation rule engine. In this case SP act as the adaptor. The CFA invocation means the situation is changed so it is fired the event which causes the adaptation. CFA is a module that receives an entity and analyzing its description is able to understand which the rules it can fire are. After, it takes the rules described in the entity from the repository. Obviously not all the rules will be fired, it depends on the situation. Thus CFA needs to receive the profile too. Once CFA has the rules and the profile, matching them will be able to understand which rule it must fire in order to return an entity adapted to SP. According to the previous section each rule consists of two parts: a condition and an action. The condition is expressed in the *when* clause as a Boolean expression using attribute of concepts which model the profile. The action is expressed in the *then* clause and consists of a function that has the same name mapped on the entity description. It is implemented in java and stored in the rule repository. When the condition is false, CFA usually doesn't do anything, but optionally there can also be a second action to perform. This solution allows us the separate rule description and rule implementation. This entails that if someone wants to change the conditions of a rule it hasn't to compile or write back the rule implementation. The actions of a triggered rule temporarily update some attributes of some concepts of the entity model. After that, entities are transformed in a Java object and returned to SP that will insert the entities in the right layout in order to allow the UI Manager to present the contents to the users.

Figure 5 and Figure 6 show two examples of adaptation.

The first is an example of a simple adaptation. The left part of Figure 5 shows the raw entities without adaptation. If a user half-blind request the same content SAPI recognise the user (for concrete adaptation even for the simple access the user is provided with an RFID tag that allow SAPI to identify the profile stereotype), retrieve the profile, analyse the context and adapts all the entities resizing each trying to use the whole page. The second example shows a more complex adaptation. A user is using a mobile phone and it isn't experienced in this service. These information are in the profile and CFA applies three rules. The first one adapts the content to the new device and since it can contain less entities it is forced to split the page in more than one. The second rule adapts the content for a user not experienced. The third is a resizing of the entities. In this case the dimension is decreased. In this example CFA fired three adaptation rules. Every entity could be adapted through more than one function. The order in which they are fired is established through a priority that helps avoiding loops. Drools allows to indicate priorities in the *salience* clause.

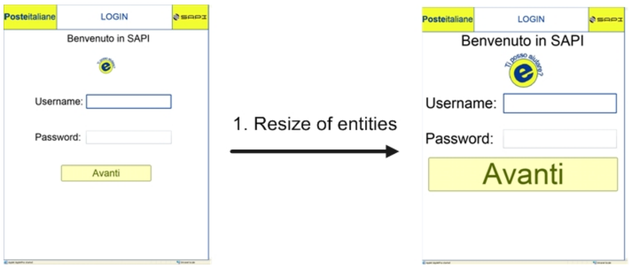
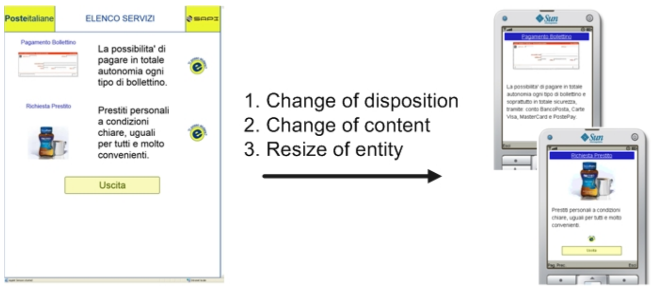


Fig. 5. Entities resize



La possibilita' di pagare in totale autonomia ogni tipo di bollettino.

Prestiti personali a condizioni chiare, uguali per tutti e molto convenienti.

La possibilita' di pagare in totale autonomia ogni tipo di bollettino e soprattutto in totale sicurezza, tramite conto BancoPosta, Carte Visa, MasterCard e PostePay

Prestiti personali a condizioni chiare, uguali per tutti e molto convenienti

Fig. 6. Contents and layout adaptation

References

1. Troiano, L., Birtolo, C., Armenise, R., Cirillo, G.: Optimization of menu layout by means of genetic algorithms. In: van Hemert, J., Cotta, C. (eds.) EvoCOP 2008. LNCS, vol. 4972, pp. 242–253. Springer, Heidelberg (2008)
2. Troiano, L., Birtolo, C., Miranda, M.: Adapting palettes to color vision deficiencies by genetic algorithm. In: GECCO 2008: Proceedings of the 10th annual conference on Genetic and evolutionary computation, Atlanta, Georgia, USA, 12-16 July 2008, pp. 1065–1072 (2008)
3. Wu, H., De Bra, P.: Sufficient Conditions for Well-behaved Adaptive Hypermedia Systems (2001)
4. RuleML Iniziative, http://www.ruleml.org
5. SWRL, http://www.w3.org/Submission/SWRL/
6. Wagner, G., Damásio, C.V., Lukichev, S.: First-Version Rule Markup Languages (2005)
7. Paschke, A., Kozlenkov, A., Boley, H.: A Homogenous Reaction Rule Language for Complex Event Processing. In: 2nd International Workshop on Event Drive Architecture and Event Processing Systems (EDA-PS 2007), Vienna, Austria (2007)
8. De virgilio, R., Torlone, R.: FAWIS: A Tool for the Automatic Generation of Adaptive Web Sites, MAIS (2006)
9. Tran, T., Cimiano, P., Ankolekar, A.: Rules for an Ontology-based Approach to Adaptation. In: First International Workshop on Semantic Media Adaptation and Personalization (2006)
10. De Bra, P., Aerts, A., Berden, B., de Lange, B., Rousseau, B., Santic, T., Smits, D., Stash, N.: Aha! The adaptive hypermedia architecture. In: Proceedings of the ACM Hypertext Conference, Nottingham, UK (2003)
11. Brusilovsky, P.: Methods and Techniques of Adaptive Hypermedia (1996)