Evolving Mashup Interfaces using a Distributed Machine Learning and Model Transformation Methodology

Antonio Jesus Fernandez-Garcia[1], Luis Iribarne[1], Antonio Corral[1], and James Z. Wang[2]

[1] Applied Computing Group. University of Almeria, Spain.
[2] The Pennsylvania State University, USA.
{ajfernandez,luis.iribarne,acorral}@ual.es
{jwang}@psu.edu

Abstract. Nowadays users access information services at any time and in any place. Providing an intelligent user interface which adapts dynamically to the users' requirements is essential in information systems. Conventionally, systems are constructed at the design time according to an initial structure and requirements. The effect of the passage of time and changes in users, applications and environment is that the systems cannot always satisfy the user's requirements. In this paper a methodology is proposed to allow mashup user interfaces to be intelligent and evolve over time by using computational techniques like machine learning over huge amounts of heterogeneous data, known as big data, and model-driven engineering techniques as model transformations. The aim is to generate new ways of adapting the interface to the user's needs, using information about user's interaction and the environment.

Keywords: Machine Learning, Mashup Interfaces, Smart Interfaces, Distributed Systems, Big Data, Model Transformation.

1 Introduction

The "smart" concept is prominent nowadays. From smart phones to smart TVs by way of classic desktop and laptop computers, the adoption of the word "smart" intends to convey the idea that the devices are no longer simple boxes manipulated by the user but have moved on so far as to add value. New concepts such as the "Internet of Things" [21], which has caused a boom in everyday devices being connected to the Internet, or "smart cities" [17], with the claim that the citizen can actively participate with its urban centre, mean that now, more than ever, devices and systems need to be increasingly "intelligent" and count with better prepared user interfaces.

That leads to today's users to require instantaneous access to information. This information is often simply raw data, but more and more frequently, it is elaborated information that must be processed in order to be offered up. The way users access information may also evolve over time due to different factors. This possible development leads to the creation of new interfaces which are capable of responding to the users' demands at the moment they are accessed.

Traditionally, computer systems are defined at the design time according to an architecture and some initial requirements. This means that in many cases, the effect of the passage of time and changes in users, applications and services is that the systems that do not adapt cannot always satisfy the user's requirements. In dynamic and evolving computer systems, where it is intended that the interface adaptation changes intelligently with the needs of users at all times, it can be useful that the implicit architecture evolves in "runtime" to adapt to user's updated needs and their environment.

We propose to evolve distributed information systems using machine learning where algorithms can be applied, analysed and evaluated in order to infer patterns of user behaviour. Through these patterns of behaviour, models which are capable of evolving the user interface at runtime can be created based on the actions performed by users and the system environment, as well as prediction models, capable of deduct users actions and propose a response (adaptation) to them. The interfaces, data and information to be analysed to generate these actions can come from different systems and distributed architectures [15].

In addition to the areas of knowledge already mentioned, the implementation of the proposed methodology for the regeneration of the user interfaces at runtime in an evolutionary way will make use of others such as cloud computing [18], big data [2] or model transformation [1]. Continued access to distributed systems and the data generated by that connection justify the use of cloud computing and big data [16]. The need to adapt the user interface at runtime to new requirements justify the use of model transformation.

The rest of the paper is organized as follows. Section 2 describes related work and the motivation. Section 3 defines in-depth the methodology for developing evolutionary systems. Section 4 explains adaptation methodology to a case study. Finally, Section 5 concludes and provides future directions.

2 Related Work and Motivation

The way people communicate with the devices around them has experienced very rapid change. Much recent work has been on the optimization of user interfaces, using different approaches. For instance, Russis *et al.* [6] focused on optimizing the interaction between users and intelligent environments. Fensel *et al.* [8] modeled interfaces which intelligently manage energy in smart homes. Roscher *et al.* [19], focused on the management of interfaces that dynamically adapt to users in their daily lives within an intelligent environment. In this regard, it is interesting to compare our proposed approach with related work. In our case, we propose the evolution of the dynamic adaptation on distributed information systems, that has not been covered in related work to our knowledge.

For that, dynamic adaptation mashup interfaces [5] are commonly used. Due to their granularity they facilitate the adaptation of their internal structure to new adaptation requirements and they make it easy to study user behavior.

Nowadays, mashup interfaces (or component-based interfaces) are widespread in commercial software, particularly in Web applications [10]. The components forming these mashup interfaces are directly integrated into the system interface according to a component architecture. The use of these components depends on the discovery and access services provided by the system itself. In this type of interface what's important is not only how the interaction occurs, but also what occurs after the interaction.

In work carried out by Iribarne *et al.* [14, 12, 9] and Criado *et al.* [4], a global method is presented in which component-based architectures are able to compose both dynamically and autonomously in order to best adapt to the user's requirements. It is worth mentioning that all the adjustments made in the system are based on a number of adaptation rules which are defined statically in a rule repository. The system for adapting the interface to the user's needs is achieved by the following processes, as illustrated in Fig. 1.

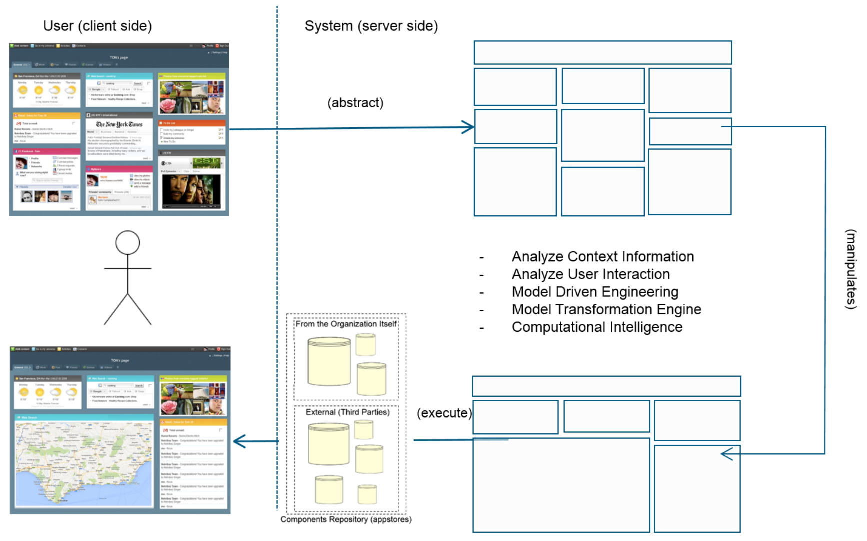

User (client side)
System (server side)
(abstract)
(manipulates)
- Analyze Context Information
- Analyze User Interaction
- Model Driven Engineering
- Model Transformation Engine
- Computational Intelligence
From the Organization Itself
External (Third Parties)
(execute)
Components Repository (appstores)

Fig. 1. The process of dynamic user interface adaptation at the runtime.

- *Abstraction of the concrete user interface.* The components which the user is using are identified, isolated and an abstract model is generated based on them.
- *Manipulation of the abstract model.* Analysing the contextual information from the model's environment, the interaction between the user and the interface and using model-driven engineering (MDE) and computational intelligence transformation of the model is produced. The transformation is performed based on the adaptation rules stored in the rule repository.

– *Creation of a new concrete model.* The abstract model is instantiated using the right components in a concrete model thus providing an interface which is better tailored to the needs of the user [14].

As aforementioned, following traditional software architecture development methods, the developed system was designed at a particular time, i.e. "the design time", based on the specific environment and needs identified at that time. The reality is that these requirements are often dynamic and can evolve in the continuous integration and deployment of the distributed system, coming to scene, for instance, with new users and components.

Consequently, it is justified to conclude that this dynamic adaptive behaviour appears to be insufficient. It would be desirable for the user interface to modify its structure at the runtime based on information from the user interaction and the distributed environment, in order to provide users with the components they require. It is even possible to make the form of dynamic adaptation evolve to the user's needs through the discovery of behavioural patterns based on the user's interaction with the interface. That would create prediction models provided by a set of adaptation rules.

3 The Proposed Methods for Developing Evolutionary Interfaces

In this section, we explain the principles on which our methodology is based. We first briefly describe the whole process, and then provide details for each step.

The general objective of this work is to propose a method which allows component based user interfaces, that comes from information system architectures suited for distributed development, to be intelligent and evolve over time. This is to be achieved through the study, analysis and the application of different machine learning algorithms, big data techniques to process large and heterogeneous volumes of data, and a model-driven methodology which allows the process of generating new forms of interface adaptation to be autonomous and continuous. This methodology for building evolutionary user interfaces is not only applicable to graphical user interfaces, but also generally extendable to all types of human machine interfaces.

Fig. 2 shows the overall process of the proposed methodology; a framework scenario which, through user interaction with the interface, context information and information on the environment, allows the creation of intelligent, dynamic and evolving interfaces in distributed systems that modify their structure at runtime.

The process gathers information on how the user handles the interface and its components together with the information about the environment. Due to the size, heterogeneity and speed of data generation, the information gathered is stored using big data techniques in data structures prepared for the purpose.

From this large volume of stored data, different data views are defined. A data view is the result set of structured data drawn from the general database

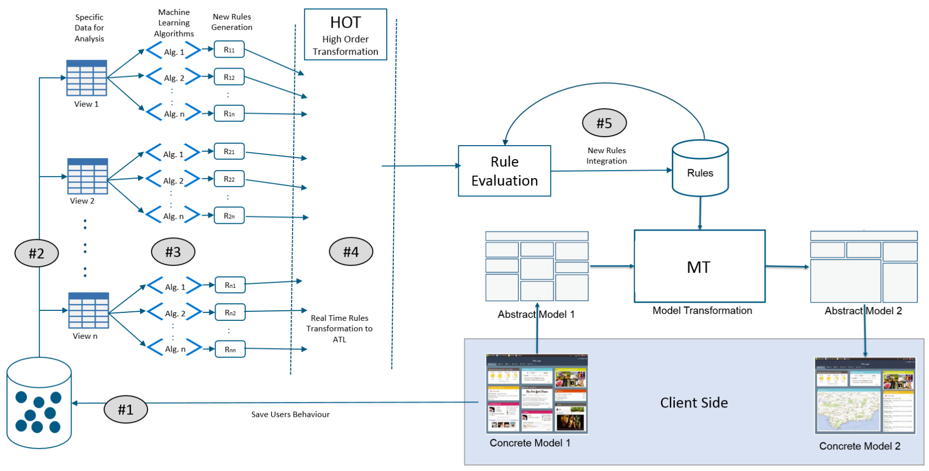


Fig. 2. Dynamic and evolutionary user interface adaptation at the runtime.

by a stored query for a specific purpose. Unnecessary information is hidden and other information which may be relevant is added.

Each of these views is focused on a type of study, analysis and evaluation through the use of algorithms and machine learning techniques. Each view is structured to apply different algorithms depending on their nature, as they can be algorithms for clustering, sorting and/or regression. The aim is to infer new knowledge which provides new adaptation rules to the system.

Once the machine learning algorithms have been applied and new rules (which provide the adaptation of the system) generated, a transformation is necessary so that the rules generated have the same format as the rules the user interface use to be adapted to the user's needs. The way the adaptation rules are written can be different for each user interface, and therefore the transformation belongs to the problem domain. Furthermore, it is necessary to define whether the new rules generated are valid or not. For this, a rule evaluation module will be implemented which includes a conflict manager.

Step 1: Distributed Storage of User Behaviour. All the activity generated by the user's interaction with the interface, as well as all the information about the environment during the time in which user interaction occurs is recorded. This information can be:

(a) Information about the user who carried out the interaction. Useful information will be stored like which particular user has made the interaction, what kind of user he is, how is he related to other users, and what characteristics define him. It will be further supplemented, if possible, with other relevant information about the environment, such as cultural information or demographic characteristics.

(b) Operations performed by the user in the interface. It should be remembered that the proposal is based on mashup type interfaces which are composed of different components. Types of operations carried out in the interface are: add a component, remove a component, move components, group components, resize components, among others.
(c) Situation where the interaction has been made. Information about the session in which the interaction is produced, such as temporal information, location information, session and interaction information, and other relevant information.
(d) Workspace status. Interface status after the performed operation. Components that are in use at the time, the position of each components relative to others and interaction between components.

Step 2: Creating Distributed Data Views. Different data views are defined on the primary database. The intended objective when creating different data views or datasets is to organise data, i.e. to structure them so that they can serve as entry elements for the algorithms used in machine learning. Data views are composed with information spread in the whole infrastructure of the distributed information systems. The structure given to data optimises the results obtained during the execution of the experiments as well as to certify that the output of the algorithms generates valid results. In other words, it maximises properly structured relevant information which is able to provide new inferred knowledge susceptible to creating new rules for adaptation of the system, thereby generating evolutionary systems.

Step 3: Application of Machine Learning Algorithms. Different kinds of existing algorithms in the machine learning literature are studied, analysed and applied to the “data views”. Experiments are carried out to test and evaluate the correct application of the algorithms, and to determine if valid knowledge can be extracted from the generated outputs. By valid knowledge, it is meant that it can generate new rules capable of improving the user interface adaptation. It is very important to study which algorithm is the most effective in each case and for each data view, depending on its nature. Two groups of algorithms can be identified:

– *Supervised.* Training data is prepared while knowing the right answers. This training data is used to build models which are able to classify new data and, in this way, obtain answers.
– *Unsupervised.* Data is directly entered into the algorithms in the hope of getting some intelligible information through the structuring of data.

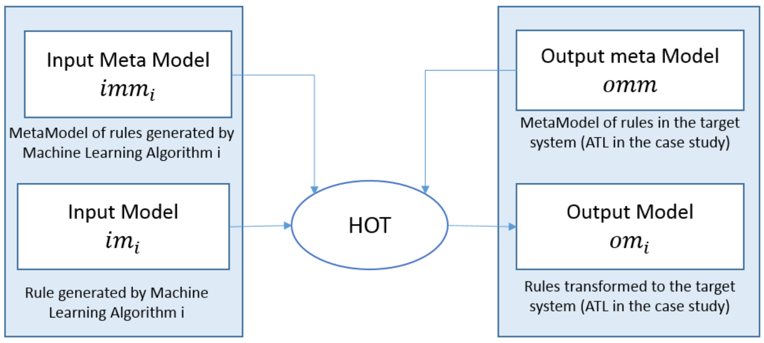
Experiments should be performed to identify which algorithm is suited to each situation and how an algorithm can be parameterized to optimize its results. Furthermore, existing algorithms can be modified to achieve the best possible results. Some algorithms of interest are: Clustering, Association Learning, Parameter Estimation, Recommendation Engines, Classification, Similarity Matching, Neural Networks, Bayesian Networks and Genetic Algorithms among others.

Step 4: Conversion Rules. The inferred rules generated by experiments with machine learning algorithms have their own format which is suited to the identity of the data processed and the type of algorithm used.

Using model transformation [13] the outputs of the algorithms are transformed to the format suited to the interface where they will be applied. The way the adaptation rules are written may differ for each user interface and therefore the transformation belongs to the problem domain.

A special kind of transformation exists in which the model transformations (i.e. the transformation models) participate as input or output models in a process called Higher-Order Transformation or HOT [20]. This kind of HOT transformation, described in Fig. 3, is used to rewrite the knowledge generated by the algorithm into an adaptation rule.

Furthermore, it is necessary to define whether the new rules generated are valid or not. For this, a rule evaluation module will be implemented which includes a conflict manager.


Input Meta Model
imm_i
MetaModel of rules generated by Machine Learning Algorithm i
Input Model
im_i
Rule generated by Machine Learning Algorithm i
HOT
Output meta Model
omm
MetaModel of rules in the target system (ATL in the case study)
Output Model
om_i
Rules transformed to the target system (ATL in the case study)

$$H(\,f(im_i, imm_i\,), omm)\colon\; om_i \;/\; \{\forall_i\colon 1 \ldots n\}$$

Fig. 3. Transformation of Rules using a HOT.

Step 5: Evaluation of Rules. The rule evaluation module will define whether a new rule generated should be added to the rule repository or not. It is possible to achieve new rules which contradict existing rules from the repository. For these particular cases, a "conflict manager" should be employed which must resolve how to handle the potential conflicts that may occur with previously established rules.

4 Case Study in the ENIA System

As a case study, the proposed methodology for creating evolutionary mashup user interfaces has been applied to ENIA [7][3] (ENviromental Information Agent),

[3] http://www.enia.dreamhosters.com

which is a component-based graphic user interface for environmental management. Different user profiles and various categories of components exist in the ENIA interface. Examples of these components include: natural areas, wetlands, drovers' roads or biosphere reserves, positioned within maps. The application of the methodology to ENIA is described below.

Distributed storage of user behaviour. The following information are stored. Operations carried out on the interface: add component, delete component, move component, resize component; User information: name, category, home region and other personal data; User type: tourist, technician, farmer or politician; Temporal information: date, time, day, month, season; Location information: location, proximity to the coast, proximity to tourist elements amongs others; Session information: duration, operations carried out; Other information: temperature, humidity, precipitation, wind amongs others; State of the workspace: location of components in the workspace, display size of each of the components.

Creating distributed data views. Storing the operations performed by a user type on a component on a certain date could be an example of a data view of ENIA. Using this data view it is possible to obtain, for example, that between January and March all farmers access the component which displays the suppliers of tomato seeds selling close to them.

Application of machine learning. There are different algorithms applied to specific cases which could be of use in ENIA. Let's assume there is a need to identify the time of day when a "tourist" user will need to find out the temperature of the water of a beach by using the "temperature of beaches" component. This component may be useful or not depending on the time of year, the home region of the user and/or their location. To resolve this type of problem it makes sense to use supervised regression algorithms. A supervised algorithm is used because we already have data of previous tourists who have used the "temperature of beaches" component and the time that it was used. These data indicate a priori the right answer to the proposed question, namely, at what time the "temperature of beaches" component was used. A regression algorithm is used because it is designed to predict a continuous value output (time of day) depending on a series of attributes (user type, season, home region or current location among others). There are many other examples where supervised classification algorithms and unsupervised clustering algorithms can be used.

Conversion rules. In order to be able to add a new rule to the rule repository, it is necessary to transform the output of the algorithm into the format of the rules stored in the repository. In this particular case we use model transformation [13] to transform the algorithm outputs and the inferred knowledge to ATL rules [11], which is the format used by ENIA. ATL is a transformation model language which also provides a number of tools for Model Driven Engineering.

Evaluation of rules. Any valid new rule generated is added to the rule repository. If a new rule contradicts an existing rule, the new rule is prioritized. The new rule is considered to have been created in a newer environment and with interactive behavioural data which has been contrasted with more users.

5 Conclusions and Future Work

A methodology is proposed to allow mashup user interfaces in distributed systems to adapt to user requirements at runtime, be intelligent and evolve over time through the use of machine learning and model transformations. The process of applying the methodology is described in the case study of ENIA, an intelligent agent for environmental information.

The creation of an automatic learning system is proposed as future work. Machine learning experiments can be performed in a way that the data is fed automatically from Web services, making the system autonomous and independent. Each new execution of an experiment will have new data collected automatically (updated datasets) which can provide different outputs.

Statistical learning techniques will be used to classify or group different user intentions for individualized interface treatment. Proper integration of learned knowledge with manually-entered preferences is an area of future development. The evaluation with human subjects will be needed to assess the effectiveness and usefulness of dynamic interfaces.

Finally, the optimization and/or improvement of algorithms is also interesting. Areas to be worked on are: defining new algorithms in extending those that already exist in the field of machine learning and customize existing algorithms in both operation and parameterization through statistical programming languages algorithms and statistical analysis as R, Octave or Matlab. It follows that these optimized algorithms should present optimum system results.

Acknowledgments. This work was funded by the EU ERDF and the Spanish Ministry of Economy and Competitiveness (MINECO) under Project TIN2013-41579-R and under a FPI Grant BES-2014-067974 and by the Andalusian Regional Government (Spain) under Project P10-TIC-6114. Wang has been funded by the US National Science Foundation.

References

1. Bollati VA, Vara JM, Jiménez A, Marcos E. Applying MDE to the (semi-)automatic development of model transformations. Information and Software Technology 55(4), 699–718 (2013)
2. Chen CLP, Zhang C. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. Information Sciences 275, 314–347. Elsevier (2014)
3. Criado C, Rodriguez-Gracia D, Iribarne L, Padilla N. Toward the adaptation of component-based architectures by model transformation: behind smart user interfaces. Softw. Pract. Exper. (2014)
4. Criado J, Vicente-Chicote C, Iribarne L, Padilla N. A model-driven approach to graphical user interface runtime adaptation. 5th International Workshop on Models@run.time, pp. 49–59 (2010)
5. Daniel F, Matera M. Mashups: Concepts, Models and Architectures (2014)
6. De Russis L. Interacting with smart environments: Users, interfaces, and devices. Journal of Ambient Intelligence and Smart Environments. 7(1), 115–116 (2015)

7. ENIA Environmental Information Agent. `http://www.enia.dreamhosters.com/`
8. Fensel A, Vikash K, Slobodanka T. End-user interfaces for energy-efficient semantically enabled smart homes. Energy Efficiency. Springer (2014)
9. Fernandez-Garcia AJ, Iribarne L. TDTrader: A methodology for the interoperability of DT-Web Services based on MHPCOTS software components, repositories and trading models. 2nd Int. Workshop of Ambient Assisted Living, (IWAAL2010), pp. 83–88 (2010)
10. Hoyer V, Fischer M. Market overview of enterprise mashup tools. Service-Oriented Computing (ICSOC'2008) Springer-Verlang, Berlin, Heidelberg. 708–721 (2008)
11. Kurtev I, van den Berg K, Jouault F. Rule-based modularization in model transformation languages illustrated with ATL. Science Computer Programming 68(3), 138–154. Elsevier (2015)
12. Iribarne L, Criado J, Padilla N, Asensio J. Using COTS-widgets architectures for describing user interfaces of web-based information systems. International Journal of Knowledge Society Research 2(3), 61–72. IGI Global (2011)
13. Iribarne L, Padilla N, Criado J, Asensio J, Ayala R. A model transformation approach for automatic composition of COTS user interfaces in web-based information systems. Information Systems Management 27(3), 207–216. Taylor and Francis (2010)
14. Iribarne L, Padilla N, Criado J, Padilla N, Vicente-Chicote C. Metamodeling the structure and interaction behavior of cooperative component-based user interfaces. Journal of Universal Computer Science 18(19), 2669–2685 (2012)
15. Mishra D, Mishra A. Distributed information system development: review of some management issues. On the Move to Meaningful Internet Systems: OTM 2009 Workshops, pp. 282–291 (2009)
16. Mishra D, Alok M. Research trends in management issues of global software development: evaluating the past to envision the future. Journal of Global Information Technology Management 14(4), 48–69. Taylor and Francis (2011)
17. Ortner E, Mevius M, Wiedmann P, Kurz F. Design of Interactional End-to-End Web Applications for Smart Cities. 24th International Conference on World Wide Web Companion, pp. 551–556 (2015)
18. Richard K, Deters L, Deters R. Architectural Designs from Mobile Cloud Computing to Ubiquitous Cloud Computing - Survey. IEEE World Congress on Services (SERVICES '14), pp. 418–425 (2014)
19. Roscher C, Lehmann G, Schwartze V, Blumendorf M, Albayrak S. Dynamic Distribution and Layouting of Model-Based User Interfaces in Smarts. In: Hussmann H., Meixner G., Zuehlke D. (eds.) Model-Driven Development of Advanced User Interfaces. LNCS, vol. 340, pp. 171–197. Springer, Heidelberg (2011)
20. Tisi M, Joualt F, Fraternali P, Ceri S, Bézivin J. On the Use of Higher-Order Model Transformations. ECMDA-FA, pp. 18–33 (2009)
21. Whitmore A, Agarwal A, Xu L. The Internet of Things - A survey of topics and trends. Information Systems Frontiers 17(2), 261–274 (2015)