ARCHITECTURE MODEL FOR PERSONALIZING INTERACTIVE SERVICE–ORIENTED APPLICATION
Cédric Joffroy, Anne-Marie Pinna-Dery, Philippe Renevier, Michel Riveill
Equipe Rainbow, Laboratoire I3S, CNRS
930, route des Colles BP 145
06903 Sophia Antipolis Cedex, FRANCE
{joffroy, pinna, renevier, riveill}@polytech.unice.fr

ABSTRACT
Service-oriented Architectures obviously improve the design of Information Systems (IS). Moreover, the omnipresence and the diversity of devices led to the development of new infrastructures which encourage the use of multiple interaction techniques to use IS. In this article, we present an architecture of evolution control which supports the adaptation of applications according to discovered services or devices, by respecting the preferences of the users as well at the level of the information system as the level of the support of interaction.

KEY WORDS
Services Oriented Architecture, Dynamic Adaptability, Multi-Device systems

1. Introduction
Nowadays, Information Systems (IS) use and development are confronted to three challenges: the increasing number of information sources, the technological rise of devices and the multiplicity of usages.

First, the increase of internet technologies brings up quantity of free information sources. Service-Oriented Architecture (SOA) is a solution to develop more easily IS allowing flexibility for the apparition and disappearance of information sources [1]. Secondly, devices for mobile usage and communicating objects are to be considered in current life. We can notice the importance of the "Ubiquitous Computing" [2] introduced by Mark Weiser: "Silicon-based information technology, is far from having become part of the environment". Considering new kind of interaction between human and computer, the interaction systems have to integrate new devices more and more original according to new interaction techniques [3]. Solutions around Web Services for Devices [4] are an assurance of flexibility to insert modern devices and interaction techniques to an open system such as IS. Third, all this new technological capabilities imply a lot of changes in the usages of applications. Very rapidly, the information diffused on papers has been replaced by electronic information accessible via computers, mobile phones or personal computers. This new kind of information supports changes the behaviour of information producers and consumers. Moreover independently of technical considerations, usage is in constant evolution following the life evolution. We are more and more mobiles as well for leisure for professional activities. We work everywhere: at our office, at home or during the transport. So the usage of IS is in mutation. Architectural solution proposed in the Human-Computer Interaction (HCI) research are interested in plasticity [5] to facilitate the adaptation of the applications to the usage and to the supports. Although those works are not based on SOA and then do not take benefits of their advantages in the building of evolutive architecture. In this paper, first we give an overview of existing solutions in the HCI domains (section 2). Then we describe an architectural solution based on the Arch Model [6] but conserving SOA advantages to personalize IS considering the multiple information sources and supports (section 3). We also show how SEDUITE [7], Polytech’Nice Sophia IS, is a good illustration of the architecture advantages (section 4). SEDUITE diffuses for example time-tables, social event and weather information. These examples are used as illustration in all the parts of the paper. To conclude, we give an assessment of our contributions (section 5).

2. Overview of HCI Solutions
From a SOA point of view, an interactive Information System is composed by a set of information services (the core of the IS) and by a set of interactive component (the interactive system). This parting is also one of the elementary principles of researches in HCI, as illustrated by the Arch Model [6]. The Figure 1 summarizes this architectural model.

The two feet of this Arch are the Functional Core (FC) and the Low Level Interaction Component (LLIC). The FC is the set of all functionalities used by the system. This is why the system is useful. The LLIC is what users manipulate during interaction. The LLIC is the visible and interactive part of the system. The keystone of the Arch is the Dialog Controller (DC). This component is why the system is usable. It adapts functionalities to output devices and information from input devices to the FC. The DC also organized the path of interactions (task model).

Looking for the modularity and reusability, two components are introduced in Arch Model between the two feet
and the keystone: the Interface with the Functional Core (IFC) and the Presentation Techniques Component (PTC). Both IFC and PTC are respectively abstraction of the FC and the LLIC. The aim is to enable evolution in one of the three main components (FC - i.e. changing implementation or algorithms of functionalities - , DC - i.e. changing HCI organisation - or LLIC - i.e. changing of devices or platform) with less changes as possible in others components.

Another research axe is dynamic personalization in HCI works: a part of plasticity [5]. By definition, the plasticity of HCI is the capacity of an interface to adapt itself to its usage context by respecting its usability. Usage context is defined as the triplet <user, environment, platform>.

We classify plasticity works on three directions:

• User Interface Language Description (UILD), e.g. W3C device independence [8] or UsiXML 1
• Model-Driven Engineering (MDE) applied to the plasticity, e.g. [9] or [10]
• Agent-Based architecture, e.g. AMF [11] or Comet [12].

Figure 1. Arch Model

In the context of multiple information sources and multiple interactive devices, SOA ensures service independence and information on assembly (orchestration). Because of the gap between the user interface and the FC, adaptations in HCI are user-centered and design-oriented, but also disconnected from the information services. More than the loose of those SOA assets, plasticity approaches are top-down, expecting a control on any manipulated entity. Results are often isolated, walled in the specific framework, rejecting any non-conform component.

Moreover, specificities of Information Systems are not especially considered. Information are shared, clients are distributed. In one hand, transformations and communication of UILD and MDE approaches could be too heavy for running on light devices (mobile phone, PDA, quickly-saturated wireless network, etc.). In the other hand, agent-based approaches require a specific development for data processing or transfer protocols, adding difficulties for enabling unexpected adaptations.

Our aim is to find a software architecture for reusing existing component and enabling adaptations sparked off by user’s choices, at any level of the Arch Model (Figure 1). So we assume that the service-oriented approach is the base of a solution. Consequently, we describe our architecture model based on service in the next section and we present its implementation with Web Service in the section 4.

3. Architecture for an adaptable IS

The overview of existing solutions shows the advantages of a solution based on SOA for interactive information systems. In this section, we describe an architectural model based on the combining of Arch Model and SOA and more specifically on an adaptable dialog controller (ADC).

3.1 Personalizing needs

Our architectural model has to allow user to personalize devices, information and the dialog controller. The user has to interact with the whole system to notify what she or he wishes; choosing the information and their representation she or he is interested in. This profiling corresponds to several kinds of preferences:

1. some of them are relative to the parameterization of the information services and their possible orchestration (level Functional Core). The user can notify its interest for the weather at Nice, for its time-table, etc.
2. some of them are relative to the filtering of the information (level Interface with the Functional Core). The user can be interested only in the weather at the morning and the evening
3. some of them are relative to the tasks sequence (level ADC). If the user is interested in several information, he can specify the consulting order
4. some of them allow to choose how to format the information (level ADC). The user can choose between a representation like flight listing in airport or like a planning for its time-table depending on its preference.
5. some of them allow to choose the preferred output devices (level Presentation Technique Component). A student can notify that he or she prefer to receive his or her results on mobile phone and consulting its timetable on the plasma screen of the school hall

1http://usixml.org/
some of them are relative to the way to the display format (font, color, etc.) (level Interaction Component).

To sum up, the ADC role is to link a task (a user objective), functionalities and the output and input. Element to adapt are: the task model, the association task-functionality or the association task-output or the association task-input.

We propose in the subsection (3.2) an architectural model allowing profiling an information system in different levels.

### 3.2 Architectural model: an Adaptable Arch

The Figure 2 illustrates the adaptable Arch combined with SOA and decorated with several adaptation elements. In the Adaptable Arch, each Arch foot (FC, LLIC) is multiple. The LLIC feet correspond to input-output devices. This Interaction Component is composed by several "Presentation Technique Component-Interaction Component" couples; each one corresponding to one Web Service for Device with its associated adaptor element. The FC feet correspond to the information services. This Functional Core is composed by several "Functional Core-Interface with the Functional Core" couples; each one corresponding to one Web Service for information and its associated adaptor.

The abstract representation of the ADC we choose is the task model. Such a model expresses the objectives (tasks) and intermediate objectives (sub tasks) a user has to reach to obtain its final target.

![Figure 2. Adaptable Arch Model](image)

The ADC has explicitly a double role of merging and splitting of data corresponding to information and interactions. The ADC has to consider the set of information sources (Arch feet of the Functional Core level) and a set of interaction supports (Arch feet of the Interaction Component level). Functional data are gathered from several sources (merging) and restructured depending on the usage (format). Then the ADC has to dispatch formatted information to the selected output devices. Interactional data are reconstituted following the interaction scheme (high level merging) to dispatch the received information to the corresponding Functional Core feet.

Not only Arch feet are composed by services, the ADC is also built with several services; each one handles a part of the adaptation and control. The ADC is based on a set of dedicated Web Services:

- a tasks service, implementation of the task model and defining which business services are required by the user and how to manage their calls (in sequence, in parallel or with fusion).
- an evaluating service, collecting the results of business services.
- a transformation service, applying a general output layout to the collected data.
- a splitting service, dispatching pre-formatted data to devices.
- an administration service, enabling users to modify preferences at any level (not only in the ADC).

The services design of the ADC facilitates its architecture evolution and adaptation depending on the orchestration of these services. Thus, functional independency of the Arch Model and the independency of services and their assemblies (SOA assets) are preserved.

The Figure 2 also integrates the adaptation elements. Those elements correspond to the preferences and are associated at each Arch level. For each Arch element (FC, IFC, PTC and LLIC levels), there are corresponding preferences (see section 3.1). So that everything is personalized from the Interaction Core to the Functional Core. The User can specify parameters of the different services' operations (FC preferences) like the country for the Weather. He can also specify where information must be displayed (DC preference), for example: Timetable in his mobile phone, and events on his laptop. And finally, he can specify some rendering information like font size, or background colour (LLIC preference).

Each foot is an independent entity with which the adaptable Arch elements interact to manage the adaptable interactive information system. Software interfaces, called manager, handle the communication between the Arch elements and its feet, the Functional Core and the interaction Component. The user interacts with an interface, called meta-interface, to give its preferences concerning the information she (or he) is interested in and their format properties. In this case, specific requests deduced from preferences extract the set of wished information.

### 4. Application: an IS for an engineering school

In this section, first we describe our case study and then we give more explanation considering the implementation aspects.
4.1 SEDUITE: a case study

SEDUITE is used in the engineering school at Nice-Sophia Antipolis University\(^2\).

The concerned population is broad: students, professors, administrative staff and visitors. Each population has its own interests depending on its role in the structure. But to interact with SEDUITE, the studied campus provide different kind of usable devices: (i) wide screens are available in corridors (Main Hall and coffee points), (ii) each user has a personal computer (laptop for professors, students and visitor, desktop for the administrative staff) and finally (iii) users often own a cell phone or a personal digital assistant.

SEDUITE is composed of a set of different services: time-table, news, etc. Users can express different wishes about using and displaying information, following their own preferences. Each kind of information is managed by a specific Web Service. The Information System could be extended by adding new services. Information are dynamic, e.g. social events service added by the Student Committee, marks service added by professors and administrative staff, etc.

SEDUITE is both in use and in development. Some features are only prototypes for now: it is the case of mobile phones. Those aspects will be integrated after debugging and validation on a large scale. But the time-table and social event are already broadcast through SEDUITE.

4.2 Implementation: a work in progress

The architecture and the communication between all Web Services were designed and implemented using WComp [13]. The WComp platform is a minimalist component-based environment for design-time and run-time. Its main properties are graphical design, Rapid Application Development (RAD) and dynamic components composition. This platform has the specificity of making easier components assembly towards a graphical designer and automatically generating events for communication. In this platform, Web Service and Web Service for Device are easily integrated, though an automatic generation of a proxy, a WComp component. We develop using C#/Net framework.

The Functional Core (FC) is implemented as a distributed set of Web Services. Information-providing Web Services are industrial products (Hyperplanning\(^3\) for timetable), free developments such as the weather service\(^4\) or student developments (e.g. social event notifications, the calendar service based on WebCalendar). The FC is in constant evolution. Timetable Web Service has been recently changed; our home-made Web Service has been replaced by a professional product (Hyperplanning) to increase timetable management by professors. The corresponding adaptor (IFC) has been quickly re-implemented and we reuse its associated preferences. The specific development is here to establish correspondence between two software interfaces: what the Web Service and the ADC expect as parameters for function calls and return as result. New Web Service are added after students’ internships or projects: we can cite the Web Service for marks broadcast for example.

LLIC is implemented via Web Services for Devices which are UPnP compliant. We mainly consider student laptops, plasma screens and Bluetooth mobile phones. PTC and IFC are also Web Service. Their role is equivalent to a bridge between Internet Web Service or Web Service for Devices and the system. Each adaptor extends a class using C# reflexivity capability - method invoke(iduser,idclient,proxy,method) - to prepare the call to the Web Service using the preferences given by the user as parameters.

The ADC is based on a set of dedicated Web Services. Our architecture model is a two way model (output – from FC to LLIC and input – from LLIC to FC). But as we have only implemented the output way, we only present Web Services in use, shown in the Figure 3.

![Figure 3. Implementation of the Adaptable Arch with Web Services](image-url)

Presentations are built from XML data which are received by the ADC from Web Services of the FC. Those data are improved according to users’ preferences thanks to XSL transformation filtering and formatting. There could be several XSL style sheets defining for example a special visual layout per device type. The appropriated XSL style sheet is deduced from user’s preferences and information given by Managers (see Figure 2). The result of the trans-

\(^2\)http://ploytech.unice.fr
\(^3\)http://www.hyperplanning.org
\(^4\)http://www.webservicex.net/globalweather.asmx
formation is a rendering abstraction in XAML\textsuperscript{5}) (eXtensible Application Markup Language). XAML is developed by Microsoft within the .Net 3.0 Framework.

XAML files are directly rendered in a Web browser. There are neither needs of compilation nor needs to generate an executable. To change of User Interface Description Language, new XSL style sheets have to be add in order to generate the right format (for example UsiXML, Flex\textsuperscript{6} or OpenJFX\textsuperscript{7} for mobile devices)

Concretely, preferences are stored in databases and are accessible through a Web Service. Therefore preferences are not dedicated to an application but to some services or usage contexts. The matching between preferences and their use (functional service call or device drive) are based on keywords. Their distribution are illustrated by the Figure 3. As default preferences, we only define three group profiles, one for students, one for professors and administrative staff, and one for visitors. The interface of personalization (META-HCI, interface used to specify preferences) is only an ASP web page, enabling users to enter their preferences.

5. Conclusion

We have presented a Service-Oriented Architecture for adaptation of interactive Information Systems. Considering this adaptation at any level of the Arch model, our contribution is to put them in correspondence and to propagate them into the Arch. Our architecture is applied to the SE-DUITE IS currently both in use and in development. The iterative definition of our architectural model is sent back and tested as one goes along.

In order to (dynamically) exploit its new capacities, the Adaptable Dialog Controller must be the main point leaning on the Managers of both sides of the Arch. On the long range, our aim is a ADC driving and dispatching adaptations of any parts (functional, dialog and interactive).

We discuss our contributions in the taxonomy of the plasticity described in [14]:

Remodelling: Modality and Abstraction. Our architectural model takes into account adaptations at any level of the Arch. Thanks to the Web Service for Devices in the WComp environment, we can change a modality for any other one.

Usage context: Environment, Platform and User. At the Interaction level, the adaptation targets the <platform, user> brace. At the ADC level, we consider users through preferences and we aim the best correspondence between available business services (FC) and available devices (LLIC). We are working on HCI composition algorithms in order to utterly automate the repercussion of evolutions.

**Meta-HCI.** Our meta-HCI is currently limited to the management of preferences. We are working of previewing preference modifications.

Automatic Learning. We did not work on this point. But we envisage it by deduction of usages in different contexts.

Redistribution. Even if we did not study redistribution in terms of ergonomics of multimodality, our architectural model enables the dispatching of HCI and users’ input, through the LLIC or the ADC, depending of preferences (for a kind of devices or because of discovering or losing of a device).

Plasticity Implementation. SOA approaches enable dynamic changes of heterogeneous services, we de facto obtain a dynamic production of HCI, the dynamic deployment of adaptations and the merging of many technological framework. The adaptation granularity is multi-scaled: changing one service by another and deeply modifying links between services, dialog and interactions. After applying adaptation, the current interaction with the IS is saved and the consistency of the usage must be guaranteed by the ADC. Finally, the localisation of adaptation mechanisms is double: inside each part of the Arch and externalized through Managers for the propagation of evolutions.

In conclusion, we presented the bases resolving completely the adaptations issue, but it remains finding the parameters automating adaptations, via the expression of the users. So further works will focus on different aspects. Firstly, we will use the profile editor to evaluate users’ profile. Our objective is to identify general profiles in order to propose a default one which is going to facilitate users preferences input. Secondly, we will study the capability to set off the profile against the usage. The objective is to decrease profile’s data. Thirdly, we will have to tackle with the problem of HCI composition in order to give adapted presentation for multiple information.

**References**


\textsuperscript{5}http://msdn2.microsoft.com/en-us/library/ms752059.aspx/

\textsuperscript{6}http://www.adobe.com/fr/products/flex/

\textsuperscript{7}https://openjfx.dev.java.net/

383


