A manifest for application generators: helping developers with the Serenoa framework

Javier R. Escolar, Cristina G. Cachón, Ignacio Marín
Fundación CTIC
C/ Ada Byron, 39 – 33207 – Gijón, Asturias (Spain)
{javier.rodriguez,cristina.cachon,ignacio.marin@fundacionctic.org}
+34 984291212

ABSTRACT
The Serenoa framework proposes an open platform for developing context-aware application Service Front-Ends. It is based on a set of languages to define user interfaces (UIs) in an abstract manner and rules to guide application adaptation. After these languages, various software modules (Runtime UI Generation Engine sub-modules) may transform the abstract application definition to final user interfaces to be rendered by actual devices for end-users to interact with it. As each of the available RUIGE sub-modules may be dedicated to the generation for specific application domains, support different types and technologies of adaptation and target different software platforms, we consider that a mechanism to help Serenoa developers to decide which sub-modules best suits their needs. The proposed mechanism is a manifest file to be provided by the developer of a specific RUIGE sub-module, which must describe the application domains that it covers, the interaction modalities supported, the hardware and software platforms targeted, the adaptation types considered (and which resources are adapted and to what other type of resource they are transformed) and any other piece of relevant information for developers to guess how good each RUIGE sub-module is for them. Additionally, a search engine (RUIGE Assistant for developers) is suggested for developers to have an entry point and, thus, be able to specify requirements and obtain a list of RUIGE sub-modules to cover their needs.

Author Keywords
Context awareness; service front-end; adaptation; user interface; search engine.

ACM Classification Keywords
D.2.2. Software engineering: Design tools and techniques. D.2.6. Software engineering: Programming environments. D.2.13. Software engineering: Reusable software. D.3.4. Programming languages: Processors. H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous.

General Terms
Human Factors; Design; Languages.

INTRODUCTION
The Serenoa framework [1] is an open platform, which intends to provide software developers with a set of software tools which collaborate among them in order to facilitate the development of context-aware Service Front-Ends. A Service Front-End is the type of application that the Serenoa framework can generate: a user interface which provides access to local or remote service whose logic is defined aside. Serenoa applications support a wide concept of context by adapting to its different aspects, such as device capabilities, user preferences and the various conditions of the environment [2].

In order to achieve the previous goals, the Serenoa framework is based on a modular architecture, as depicted in Figure 1.

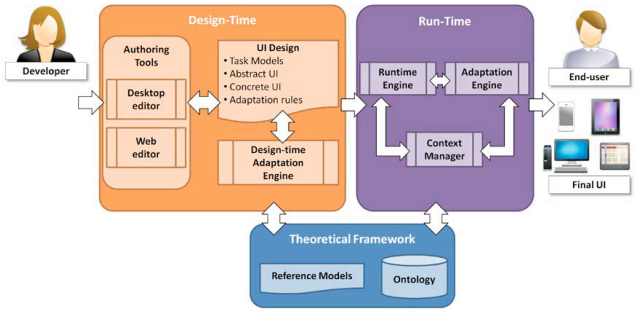

Figure 1. Serenoa architecture (as depicted in Serenoa deliverable D.X.X)

The main idea on which the framework is based is the definition of applications at a high abstraction level by means of a language called AAL-DL (*UI Design* module of the diagram in Figure 1), and a language named ASFE-DL to express adaptation rules under the Event-Condition-Action paradigm (used in the *Design-time Adaptation engine*). Afterwards, the abstract application definition is processed by the RUIGE (Runtime User Interface Generation Engine) module, which corresponds to the *Runtime Engine* module in Figure 1. This module is actually composed of different sub-modules, each of them

in charge of the generation of the an abstract application definition for different target platforms, domain applications, audiences, or supporting different types of adaptation or interaction modalities, for instance. It is not necessary that all the RUIGE sub-module have to be operated at the same time for a single application by a developer. Actually, a typical situation may be that a developer decides to work with only one or two RUIGE sub-modules.

In order to facilitate the use of the framework by the developer community, a set of assistive tools is provided. The first available set of tools is focused on application design by means of a web IDE to create and refine both the specification of user interfaces (AAL-DL language) and of adaptation rules (ASFE-DL language). We consider that more work needs to be done in the creation of assistive tools for developers using the Serenoa framework. At this moment, there are several RUIGE sub-modules, such as MARIAE (for desktop and mobile platforms and supporting multimodal interaction), UsiXML (also focused on multimodal multi-device applications) or MyMobileWeb (for mobile web browsers), an avatar-based module (in which the user interface is an avatar of a human being, supporting voice and gesture interaction), LEONARDI (specialized in business applications for native and web clients on desktop and mobile platforms, in which the UI must adapt to changes in the data model), and the warehouse module (focused on multimodal interaction in a specific scenario: operators in a warehouse). The number of RUIGE sub-modules created so far already suggests the need to present their characteristics in an organized manner for application developers evaluating the Serenoa framework. For example, there are RUIGE sub-modules specialized in specific application domains (LEONARDI, for business applications; or the warehouse module, for the interaction of warehouse operators with their environment). Some of them support resource (images, video, audio) transcoding, but there is no formal specification about which source and target formats are supported. These are clear examples of information that developers intending to use the Serenoa framework will need to know, with the only chance to read an important amount of documents or even having to contact the developer of the different RUIGE sub-modules. This problem will become increasingly relevant as new sub-modules are added to the framework over time.

Therefore, we propose that each RUIGE sub-module is annotated with a manifest file that indicates all the information that may be relevant for application developers in order to decide whether each sub-module is sufficiently good for their purpose. In addition, we consider that gathering the manifest descriptions of all the RUIGE sub-modules in a repository, and allowing queries to this repository after the application requirements provided by developers, is an interesting assistive tools. In this way, developers may know in advance whether one or more of these sub-modules facilitate the creation of the application that they want to implement.

The following paragraphs discuss the idea previously proposed in the different sections of this document. After this introduction, Section “Manifest file definition” proposes the initial ideas for a definition of the manifest file. Section “RUIGE assistant for developers” suggests the fundamental concepts on which the search engine assistive tool, which matches application requirements and RUIGE sub-module features, would be based. Finally, Section “Conclusion and future work” states the conclusion of this work and advises new ideas for consideration in order to refine the initial idea in this document.

MANIFEST FILE DEFINITION

The different concepts to be expressed in the manifest file, as a means to describe the distinct RUIGE sub-modules may be found in the Context-Aware Design Space (CADS) and Context-Aware Reference Framework (CARF), defined in [2] as part of the work of the Serenoa project.

The CADS is a theoretical method that provides stakeholders a tool to support them in the phases of implementation, analysis and evaluation of adaptive and adaptable applications. The goal of CADS is helping developers before implementing their applications to be aware of possible dimensions and granularity levels for performing adaptation, and after the implementation to analyse, evaluate and compare these dimensions regarding their respective coverage levels. As such the CADS supports the analysis and the comparison of different applications that execute adaptation and during their complete development lifecycle. The concepts in CADS are categorized in Meta-UI support level, level of adaptation, UI component granularity, state recovery granularity, UI deployment and technological space coverage.

The CARF is defined as a reference framework that specifies the most relevant concepts to implement and perform context-aware adaptation. This reference framework has a graphical representation composed by seven branches that contains potential instances for implementing, performing and also analysing context-aware adaptation. It may be represented by means of a mind-map, with seven branches indicating the seven abstract concepts involved in context-aware adaptation:

- **Why**, defining the main goals for the adaptation process. For instance, adaptation may be performed to save battery consumption at client-side.
- **What**, which describes the type of resources that may be adapted, including presentation elements, media resources such as audio or video, and navigation flow.
- **Who**, referring to the actor who triggers, manages or executes an adaptation process.

- **When,** which represents the state when adaptation takes place (e.g., design time, run time, or compilation time).
- **Where**, which indicates the virtual location where the adaptation happens. Some examples of virtual location are client side, server side or an intermediate proxy.
- **To what**, reflecting context information that justifies and defines the adaptation process. For example, colour adaptation to improve the contrast for *users with low-contrast vision*.
- **How**, defines how the adaptation process is performed, including the methods, strategies and techniques used.

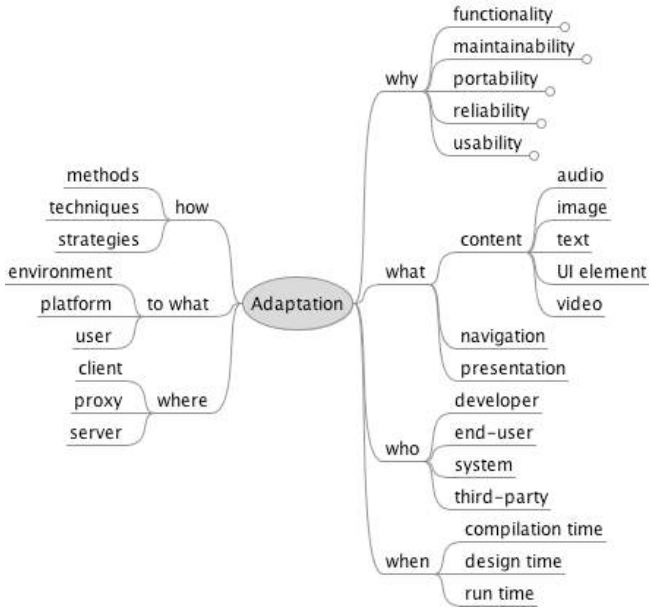

Figure 2. Example of CARF instance with the most relevant concepts for context-aware, as shown in [2]

The proposal in [3] to reflect CARF as a mind map has been considered by the authors of this article as a good starting point in order to represent the manifest file. This conceptual model may be easily translated as an XML file representing the tree structure associated to a mind map, or even as a JSON file. More information required, in order to add information from the CADS, would be added following a tree structure by adding a root node, ancestor to the adaptation root node in the CARF mind map, and then adding CADS elements.

Leaving aside the manner in which the file (or an alternative physical representation for the logical tree structure) is formatted, the main problem to develop the manifest file is the large amount of vocabularies required in order to express all the information. This may be minimized after an analysis of existing vocabularies. For instance:

- **Internet Media Types [3],** also known as MIME Media Types [4], are the elements included in the listing curated by the IANA (Internet Assigned Numbers Authority). These elements are a two-part identifier for resources available on the Internet. They are widely used in Internet protocols, such as SMTP, POP3, IMAP, HTTP, RTSP, RTP or SIP. It does not only cover content, such as audio, video, text, images, animations, etc. (thus allowing, supported formats by a RUIGE sub-module –for instance, as source and target formats of an adaptation process), but also messages and application formats. The latter is very interesting in order to express, for example, the format in which an application is generated by a RUIGE sub-module –namely, "application/x-apple-diskimage" for a Mac OSX disk image, or "application/x-winexe" for a Windows binary executable.
- **UAProf vocabulary [5],** which allows the description of software and hardware platforms, as required (for example), in the "to what" branch of the CARF. Under a more fine-grained CARF description, it also allows the reference to hardware parts or software modules of an operating system, including for example Bluetooth profiles, radio communication technologies (such as GPRS, EDGE, UMTS, HSDPA, and LTE).
- **W3C Ontology for Media Resources [6],** which defines a core vocabulary for description of media resources, and their mapping to elements from a set of existing metadata formats. For instance, it supports the definition of the author, creation/edition date and location, description, keywords, genre, rating, relation with other resources, copyright and policy information, fragment identification, compression, format, and other metadata about each resource. This opens more powerful expressivity in the "what" branch, for instance.
- **Ontologies for user description,** with a large amount of research efforts devoted to this topic in the last decade, such as "Creating an Ontology for the User Profile: Methods and Application" [7], FOAF [8] or SIOC [9].

These and other vocabularies, identified after a more rigorous analysis of the state-of-the-art may provide support to a great part of the complete vocabulary required to express CARF and CADS properties. It must be also taken into account that some other vocabularies would be simple enough so a set of values would be sufficient for each element in the vocabulary, as it can be seen for the "when" or "where" branches in Figure 2 –at least for a non fine-grained first approach. As commented before, a vocabulary describing application domains would be required in order to express business limitations for some RUIGE sub-modules, such as "applicable only to warehouse picking scenario" or "applicable only to business applications", or technological limitations, such as "targeted at mobile web browsers".

RUIGE ASSISTANT FOR DEVELOPERS

After each RUIGE sub-module is described by means of a manifest file, a possible extension for Serenoa would be the inclusion at the main page of the project of a link to what we have called the "RUIGE assistant for developers". The

idea behind this assistant is that a developer facing the challenge of creating context-aware applications with Serenoa can easily discover the best RUIGE sub-modules.

This assistant would be a web application in which developers would set the main features of the applications that they want to implement with Serenoa, checking them against the information contained in the manifest file for each RUIGE sub-module existing in the framework. Developers would be advised for each feature with their accepted values by means of a combo box or text auto-completion.

- **Application domain/business domain,** so the results returned by the assistant would recommend RUIGE sub-modules for that specific domain. Alternatively, it would also recommend those with no specific application/business domain declared, considering that those are generators of any type of application –but sorting them later in the list of matching RUIGE sub-modules.
- **Resource adaptation formats desired** would be, in its simplest form, two lists including the Internet media types accepted as source and target for adaptation process. More advanced options would support different source/target pairs, one per resource type: audio, images, video, etc.
- **Audience**, so developers could express the set of users targeted. For instance, colour-blind users.
- **Target devices**, in order to express software and hardware platforms to be covered, including the distribution format for the application –for instance, a Flash application or an executable binary.

In general, developers should be able to query which CARF and CADS features are supported by each RUIGE sub-module and thus decide whether they match the requirements for the application in scope.

Several questions may arise from this simple first approach based on syntactic search, such as what to do when a RUIGE sub-module matches the business domain required by a developer, but another RUIGE sub-module (not specialized in a single business domain) targets all the devices desired by the developer.

CONCLUSION AND FUTURE WORK

The authors believe that an entry point clarifying the possibilities of all the available RUIGE sub-modules to the developer community is an interesting issue that needs to be covered in some way by the Serenoa consortium. The application of more complex information models to express the manifest description would highly improve the functionality of the RUIGE assistant for developers. For instance, modelling the CARF and CADS in the CARFO ontology would allow that the manifest was expressed by means of RDF triples which would refer the entities expressed in CARFO. By defining the appropriate relationships, and perhaps adding a rule set to implement the corresponding decisions, expressivity problems deriving from the simplest syntactic search approach to implement the assistant would be solved –for instance, to balance different criteria as in the conflict expressed at the end of the previous section.

ACKNOWLEDGEMENT

This work received funding from the European Commission's Seventh Framework Program under grant agreement number 258030 (FP7-ICT-2009-5).

The authors wish to thank the participants in the Serenoa project for the cession of material for this article.

REFERENCES

1. Serenoa project. http://serenoa-fp7.eu.
2. Motti, V. G. D2.1.2. CARF and CADS. Public deliverable of the FP7 Serenoa project. 2012.
3. Bray, T. Internet Media Type registration, consistency of use. World Wide Web Consortium. http://www.w3.org/2001/tag/2002/0129-mime.
4. Internet Assigned Numbers Authority. MIME Media Types. http://www.iana.org/assignments/media-types/index.html.
5. The Open Mobile Alliance. WAG UAProf (2001). http://www.openmobilealliance.org/tech/affiliates/wap/wap-248-uaprof-20011020-a.pdf
6. Champin, P. et al. Ontology for Media Resources 1.0. W3C Recommendation. 2012. http://www.w3.org/TR/2012/REC-mediaont-10-20120209/.
7. Golemati, M. et al. Creating an Ontology for the User Profile: Methods and Applications. In Proceedings of the First International Conference on Research Challenges in Information Science. 2007.
8. The Friend Of a Friend project. http://www.foaf-project.org.
9. The Semantically-Interlinked Online Communities initiative. http://sioc-project.org.