Mobile AAC Services Development Process: From Usability Requirements to Reusable Components

Ivan Vučak*, Hrvoje Belani**, Marin Vuković***

* Sedam IT, Koledovčina 2, HR-10000 Zagreb, Croatia  
** Croatian Institute for Health Insurance, Margaretska 3, HR-10000 Zagreb, Croatia  
*** University of Zagreb, Faculty of Electrical Engineering and Computing, Department of Telecommunications, Unska 3, HR-10000 Zagreb, Croatia  
ivan.vucak@sedamit.hr; hrvoje.belani@hzzo-net.hr; marin.vukovic@fer.hr

Abstract. Intensive development of information and communication technologies can contribute greatly to enhancements in alternative and augmentative communication (AAC) for individuals who experience difficulty communicating in an understandable manner. Availability of Internet infrastructure and affordability of mobile computer devices provide a sound starting ground for planning and building mobile AAC services. In order to make the services more familiar and understandable to users, it is justifiable to build them on top of common service platforms in an adaptive and configurable manner. In this paper we present a component-based development process that enables building such services by using mandatory AAC components on top of the component framework, and selecting visual or non-visual components from the available component pool. The AAC component selection criteria are based on matching the data and metadata of specified functional and usability requirements with the metadata of available components from the component pool. Registering new components into the reusable component pool extends the pool and helps building the component-based AAC services development platform.

Keywords: alternative and augmentative communication, component-based development, usability requirements, component pool, AAC services.

1 Introduction

The means of communication in today’s interconnected world are more diverse than ever. Continuous growth, availability and affordability of information and communication technologies (ICTs) are considered crucial for bringing more people into information age [1]. New ideas in information exchange and content production offer new and transformative applications and services [2]. Raising popularity and affordability of mobile computers, like tablets and smartphones, along with the common mobile phones and notebook PCs, provides an access capability for various growing and emerging user groups to different ICT services. Nevertheless, in order for these services to be fully accepted and utilized by the users, along with the proper functionality they have to offer, their priory need is to be usable and trustworthy.
In order to make the services more familiar and understandable to users, it is justifiable to build them in an adaptive and configurable manner, therefore relying on software reuse. This is exactly one of the characteristics of component-based software engineering (CBSE), which aims at creating software systems and ICT services faster at lower costs and with increased stability. A software component is a small independent unit of composition with contractually specified interfaces and explicit context dependencies only. It can be deployed independently and is subject to composition by third parties [3]. In order to build a component-based system, as any other software system, the corresponding requirements specification has to be fulfilled, i.e. the requirements implemented. There are various requirement categories and types, along with their interdisciplinary nature, which brings complexity into narrowing the requirements-implementation gap. This is the main reason why we limit the requirements scope in the paper to usability requirements, which are of crucial importance to implementation of ICT services for enhancing alternative and augmentative communication (AAC). Therefore, we propose a new AAC service development process with the underlying component model, which consists of five steps: (1) gathering requirements from the users, (2) specifying requirements with specialized tool(s), (3) finding existing components with the required functionalities, (4) developing new components with remaining functionalities if necessary, and (5) implementing communication between components.

This paper is organized as follows: the next section elaborates requirements engineering for AAC services development. The third section provides an approach of component-based software development to AAC field, proposing an AAC service development process that utilizes it. The fourth section analyses the non-functional requirements and architectural components interdependency inside the proposed process, providing and elaborating an AAC service example. The last section gives conclusion, along with the comments on platform-based AAC service development and future work plans.

2 Requirements Engineering for AAC

Both for ICT services and the underlying devices themselves, usability can be defined as an extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use [4]. Augmentative and alternative communication is “the supplementation or replacement of natural speech and/or writing using aided and/or unaided symbols” where “use of aided symbols requires a transmission device” [5]. It is a field of study concerned with providing devices and techniques in order to augment the communicative abilities of individuals who experience difficulty speaking, writing and other skills of communication in an understandable manner [6]. As stated in [7], the main goal and purpose of all assistive technology for AAC is to provide individual user with the ability to generate meaningful phrases and express his/her feelings, thoughts, needs and desires. Depending on the individual’s skills and abilities, the decision has to be made which device to use, among various devices that contain different features and different levels of complexity.
Applying the same statements to ICT services for enhancing AAC, it is evident that various services have to be offered, in order to fulfill different user requirements and needs. However, having a great number of various services on different platforms, without an efficient way of precisely choosing the desired service features and properties, is not contributing effectively to the targeted user group. In this paper we state the need for introducing a development process for developing AAC services in a configurable and adaptive manner. It involves building and updating the library of reusable AAC components, both visual and non-visual, which can be chosen for building a new AAC service regarding the specified usability requirements.

2.1 Usability Requirements Taxonomy

The requirements for a system can be described as descriptions of the services provided by the system and its operational constraints [8]. They reflect the needs of customers for a system that helps solve some problem or answer on demand, such as controlling a device, placing an order or finding information. In a software industry over several decades, requirements were considered as labels for abstractions used to carry the value stream into development and on to delivery to customers [9]. Practically all existing requirements engineering (RE) paradigms put customers in central place, but most of the mainstream RE methods and techniques, such as surveying [10] and interviewing, are not completely suitable for immediate users in our case, the persons with disabilities and special needs. Regardless the usage of traditional or agile RE approaches [9]; we recommend these techniques to be carefully adopted for the customers and users that are not typically considered during the ICT service design process. Also, techniques should actively include people who support these users, such as caretakers and guardians. Diversity of user population should be analyzed, as well as a wide spectrum of functional (FR) and non-functional requirements (NFR) that should be implemented under the same service.

For the purpose of this paper, we have adopted the usability requirements taxonomy to serve as a starting point for requirements specification in our AAC service development model. In a broader sense, usability requirements, given in Table 1, should describe the means of perceived usefulness and ease of use for the specified user interface, and the service as a whole.

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirement type</th>
<th>Requirement type description</th>
</tr>
</thead>
<tbody>
<tr>
<td>URT1</td>
<td>Accessibility</td>
<td>Ease of access to and use of specific functionality.</td>
</tr>
<tr>
<td>URT2</td>
<td>Aesthetics</td>
<td>Aesthetics of the user interface and description of „look and feel“</td>
</tr>
<tr>
<td>URT3</td>
<td>UI consistency</td>
<td>Consistency of the user interface, both within the system and with other systems.</td>
</tr>
<tr>
<td>URT4</td>
<td>Ergonomics</td>
<td>Aspects of the user interface, such as avoiding unnecessary clicks, uncomfortable touchscreen moves, etc.</td>
</tr>
<tr>
<td>URT5</td>
<td>Ease of use</td>
<td>Ease of learning and using the system.</td>
</tr>
</tbody>
</table>
2.2 Requirements Interchange Format

While natural language requirements themselves are too demanding to be managed efficiently and their implementation status traced throughout the development process, it is advisable to provide a tool with an appropriate, machine-processable way of specifying and storing them. For the purpose of this paper we have chosen to specify requirements in so-called Requirements Interchange Format (ReqIF) by using the open-source tool ProR [12].

ProR is Eclipse-based tool for requirements engineering that supports the emerging RIF/ReqIF standard [13] that provides interoperability with other authoring requirements management tools in industry. ReqIF is an XML-based file format used to exchange requirements, along with its associated metadata, in an open, non-proprietary and tool-independent manner. One of the main ReqIF features is hierarchically structured specification of uniquely identified requirements that have associated attributes to them and established relations between them. The main vision of ProR is to provide reliable traceability between natural language requirements (NLRs) and formal models [12], namely in Event-B notation [14].

3 Component-Based Software Development

Although CBSD is not a novel approach, it is considered a good way to increase cost efficiency in software development and also provide increased reliability of the software when up and running, while decreasing pressure on maintenance [3]. CBSD consists of two main parts: (1) component architecture and (2) component-based development procedure. The component architecture acts as a standard foundation for the reuse of software components [15]. If architecture is standardized than the development process is able to use components as a central aspect [15].

Component-based architecture design pattern consists of components, component types, component model and component framework. Components are software implementations with one or more interfaces and component types are platform distinct types that have specialized role. Component model is made of component types, interfaces of these types and rules of interaction between types. Finally, component framework provides a variety of runtime services to support component model and acts like operating system with special purpose to provide support for components.

3.1 Component-Based AAC Software Development

In this paper we are proposing component-based model for developing AAC systems, and for that purpose we have designed AAC component model and specified AAC component types. Also we have designed specific component framework for our custom AAC components. Framework acts like container to all components and provides ability for components to co-exist. There are some components that are
mandatory, such as e.g. “AAC symbol provider”, so we can observe them as framework functionalities because every other component must interact with some of these. Framework is responsible for interaction with device (through device operating system), while regular (non-mandatory) components are not allowed to interact with the device directly. If there is a need for using the device specific features, there should be new mandatory component, providing an interface for using that feature.

There are three different AAC component types: (1) mandatory component type, (2) visual component type, and (3) non-visual component type. Difference between these three types is in interfaces that they are implementing. Mandatory component type components are implementing functionalities that are necessary for AAC service to work. Mandatory components are:

- **Symbol component** – provides other components with the ability to fetch and work with symbols used in system,
- **HCI component** – provides other components with interfaces for user interaction through human computer interface (HCI),
- **Notification component** – used for broadcasting event notifications between components,
- **Registration component** – used for gathering data about component states.

Framework must be aware of all components and their particular state.

As shown in Figure 1, mandatory components are directly interacting with the framework and they are providing interfaces for other components.

![Figure 1. AAC framework with mandatory components](image)

Visual type components have the ability to interact with user by presenting graphical elements on device interface while non-visual components are used for implementing non-interactive and non-visual functionalities.

Beside mandatory components, additional components should be available for building new AAC services, depending on the target application. These components are expected to have common functionalities such as: selecting a symbol, fetching a symbol metadata, sending a symbol based message, symbol-to-text and text-to-symbol transformation, etc.
New AAC service development process (Figure 2.) with proposed component model, consists of 5 steps: (1) collecting requirements from user, (2) specifying requests with ProR, (3) finding existing components with required functionalities, (4) developing new components with remaining functionalities and (5) implementing communication between components.

![Figure 2. Proposed AAC service development process](image)

Once the user requirements are gathered and formally specified in ReqIF format a XML notation of component requirements is used as an input to the component model. It is then determined whether the current model implements all required functionalities or part of them should be implemented as new component(s).

We claim that this process, with the reuse of developed components, provides faster development of new AAC services with no need for “reinventing the wheel”.

4 Mobile AAC Service: A Case Study

This section describes the design and implementation case study for a symbol-based eEmail client for people with complex communication needs. Examples of NLRs are presented in Table 2 in order to illustrate functional and usability requirements for this service, with regards to requirement types proposed in section 2.

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirement title</th>
<th>Requirement description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR1</td>
<td>Symbol selection</td>
<td>User uses symbols, selecting them one by one, to compose an e-mail message he/she wants to send.</td>
</tr>
<tr>
<td>FR2</td>
<td>Recipient selection</td>
<td>User selects the relevant recipient (e.g. mother, brother, etc.) of the message from a predefined group of recipients (e.g. family, teachers, etc.).</td>
</tr>
<tr>
<td>FR3</td>
<td>Message sending</td>
<td>User sends a symbol-based message to the defined recipient by choosing the “Send” option (also presented with an adequate symbol on GUI).</td>
</tr>
</tbody>
</table>
Table 2. Functional (FR) and usability (UR) requirements from the case study (continued)

<table>
<thead>
<tr>
<th>No.</th>
<th>Requirement title</th>
<th>Requirement description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FR4</td>
<td>Category selection</td>
<td>User has a possibility to choose from different symbol categories, in order to describe more complex expressions, like cheer, sadness, etc.</td>
</tr>
<tr>
<td>UR1</td>
<td>Position of the screen</td>
<td>User has a possibility to choose which position or positions of elements on the screen are the most accessible to him/her (URT1: Accessibility).</td>
</tr>
<tr>
<td>UR2</td>
<td>Symbol gallery</td>
<td>User has a possibility to choose which set of symbols to use from a gallery of available symbols (URT2: Aesthetics).</td>
</tr>
<tr>
<td>UR3</td>
<td>Size of GUI elements</td>
<td>User has a possibility to choose from different sizes for GUI elements (URT3: UI consistency).</td>
</tr>
<tr>
<td>UR4</td>
<td>First entry field focus</td>
<td>When a symbol-based dialog box is opened, the focus shall be on the first entry field, in the top position of the dialog box (URT4: Ergonomics).</td>
</tr>
<tr>
<td>UR5</td>
<td>Help options</td>
<td>The application has to offer a “Help” menu with proper instructions for users (URT5: Ease of use).</td>
</tr>
</tbody>
</table>

After configuring (labels, data types, requirement types, etc.), customizing and using the ProR tool to gather and specify service requirements into the ReqIF project, the given requirements specification can be exported as the corresponding ReqIF file. The file is used as an input for selection process from the AAC component pool. Besides the description, every requirement has a certain number of attributes that specify it in more details, e.g. requirements source, creation date, owner and status.

Some requirements’ attributes can serve as keywords for the search through the AAC components metadata, in order to find a suitable component candidate to be included in the further selection process. The data and the metadata from the ReqIF file can be matched with the metadata describing available AAC service components, helping developers decide to include it into the service or not. Figure 3 shows ReqIF requirements specification and component definition of an example component in component pool used for sending eEmail with symbols.

Left side of the Figure 3 shows a part of the specification for functional requirement FR1, which describes user selecting symbols in order to compose an e-mail message he/she wants to send. The requirement is more specified with attributes (added and configured through the ProR tool by a person – requirements analyst), which enables us to compare them against the pool of available AAC components. Right side of the Figure 3 shows the metadata description of one component from the component pool, which functionalities include sending symbol-based e-mails.

Matching the requirement attributes (from ReqIF specification) with the component metadata on functionalities (from the component pool) results in one or more component candidates for further analysis and possible inclusion in the new AAC service.
In described case study all functional requirements (FR1 - FR4) are already implemented in components and they are be found in component pool. However, only three usability requirements (UR1, UR2 and UR3) are developed and can be used in building case study service while others should be implemented.

After all required functionalities are implemented (within new or existing components) components should be integrated into service by using the framework functionalities. Framework serves as a glue to connect components and to provide mechanisms for inter-component communication. Beside functionalities, framework provides application wrapper for starting and running the service.

Described case study shows possibilities that can be achieved with the proposed AAC service development process. Service is developed with consideration of user requirements and previously developed components were used during the development process. This approach resulted with new service that fully meets user requirements and rapid development process.

5 Conclusion and Future Work

For building a shared set of service functionality and software assets using a common means of production, software systems engineering introduced the area of software product lines in the last few decades [16]. Requirements engineering for platforms is a discipline full of challenges [17], such as incompleteness of non-functional requirements, and a large variety of users with different and complex communication needs.

The first challenge can be tackled by an iterative approach while building requirements specification, with continuous inputs from the user representatives, and, in more cases, their everyday caretakers, or close relatives. The second challenge can be technically bounded by building component libraries with the components specialized for one aspect of special need (e.g. impaired hearing), but including the
expertise from other scientific fields, like psychology and rehabilitation, remains
crucial here. Nevertheless, some platform models for symbol based communication
services are already being introduced [18] as a result of multidisciplinary
collaboration [7], aiming towards the full development of platform-based AAC
services.

Acknowledgement

This work was carried out within development project "ICT systems for Persons with
Complex Communication Needs", supported by University of Zagreb Development
Fund.

References

Addison-Wesley, Boston, USA (2002)
4. ISO 9241-11:1998(E): Ergonomic requirements for office work with visual display
terminals (VDTs) – Part 11: Guidance on usability. International Organization for
5. Lloyd, L.L., Fuller, D.R., Arvidson, H.H.: Augmentative and Alternative Communication:
Evolving Technologies and Emerging Applications. Taylor & Francis Group, New York,
USA (2008)
7. Stančić, Z., Frey Škrinjar, J., Ljubešić, M., Car, Ž.: Multidisciplinary collaboration and
ICT services for people with complex communication needs. In: Proceedings of the
MIPRO 2011 / MEET and GVS, Rijeka, Croatia, pp. 265—270 (2011)
8. Sommerville, I. Software Engineering. Addison-Wesley & Pearson Education (9th
9. Leffingwell, D. Agile Software Requirements: Lean Requirements Practices for Teams,
Programs, and the Enterprise. Pearson Education, Boston, MA, USA (2011)
Requirements Elicitation. In: Proceedings of ConTEL 2005, Faculty of Electrical
Engineering and Computing, University of Zagreb, Croatia, pp. 465—469 (2005)
0-321-38300-6, Pearson Education & IBM Press, Indianapolis, USA (2007)