Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 1157–1162

DOI: 10.15439/2015F259
ACSIS, Vol. 5

Run-time UI Adaptation in the Context of the Device-Independent Architecture

Jacek Chmielewski
Poznan University of Economics
Department of Information Technology,
Al. Niepodległości 10,
61-875 Poznan, Poland
Email: chmielewski@kti.ue.poznan.pl

Abstract—**The increasing diversity of end-devices used by users to access their applications and systems strengthens the need for device-independent methods for implementing these applications. The Device-Independent Architecture (DIA) is one of the available approaches to this problem, but it does not directly address the issue of user interface (UI) device-independency. This issue can be addressed by real-time UI adaptation, but it is not clear whether the DIA architecture requires new UI adaptation methods or may use existing ones. This paper presents results of our analysis of this issue. Through theoretical model-based analysis of UI adaptation in various application architectures and through case studies of practical UI adaptation solutions we came up with a conclusion that the DIA-based systems may use existing real-time UI adaptation methods. Although, they have to be used with a different set of optimization criteria.**

I. Introduction

THE development of software applications that use end-devices to communicate and interact with users becomes a complex and time-consuming issue. The increasing diversity of Internet-connected end-devices (especially mobile devices) forces application developers to implement multiple variants of each application. Each software platform (Windows, Android, iOS, etc.) and each device type (smartphone, tablet, laptop, watch, glasses, smart TV, etc.) has its own requirements and constraints, which makes it difficult to address all of them with a single uniform implementation. Device-independency of the application logic and data is hindered by different programming languages and disparate APIs provided by different software platforms. Device-independency of the application user interface (UI) is even harder to address because of the number and diversity of possible input and output user communication channels – starting with screen sizes and resolutions and ending with non-standard symbolic interfaces popular in Internet of Things solutions.

To cope with this problem we have proposed the Device-Independent Architecture (DIA) [1] which solves the logic and data device-independence issues. However the DIA does not directly address the UI device-independence aspect, which is supposed to be solved with proper UI design [2], [3] and UI adaptation [4], [5].

To make sure the DIA does not hinder the ability to use UI adaptation to provide UI device-independence in the reported research we have sought to answer the following question: **Does DIA-based software may use existing real-time UI adaptation methods?**

To be able to properly analyze the problem we have defined a model of the run-time UI adaptation and generation process. We have used this model to theoretically examine the run-time UI adaptation and generation process in various software architectures similar to the DIA. Additionally we have performed a series of case studies of real implementations of UI adaptation methods to check if these practical solutions confirm our theoretical conclusions.

Our main findings are the following. Through our research we have shown that DIA-based software may use existing real-time UI adaptation methods designed for client-server systems. Moreover, we have learned that the main limiting factor for DIA-based implementations of these UI adaptation methods is not the performance of an end-device, but network latency and throughput. Therefore, to provide properly optimized UIs for DIA-based solutions, existing real-time UI-adaptation methods have to be used with a different set of key metrics and guidelines.

The paper is composed of five sections. Section I is the introduction. Sections II and III provide background information on the topics of UI adaptation and Device-Independent Architecture. Section IV contains the main discussion and overview of case studies. Finally, the paper is briefly concluded in Section V.

II. UI adaptation

UI adaptation activities can be split into two phases: design-time UI adaptation and run-time UI adaptation. These two UI adaptation phases focus on different aspects that may influence the UI adaptation process. The whole process, with its various aspects, is best described by the CAMELEON Reference Framework [6], which provides designers and developers with generic principles for structuring and

This work was supported by the Poznan University of Economics

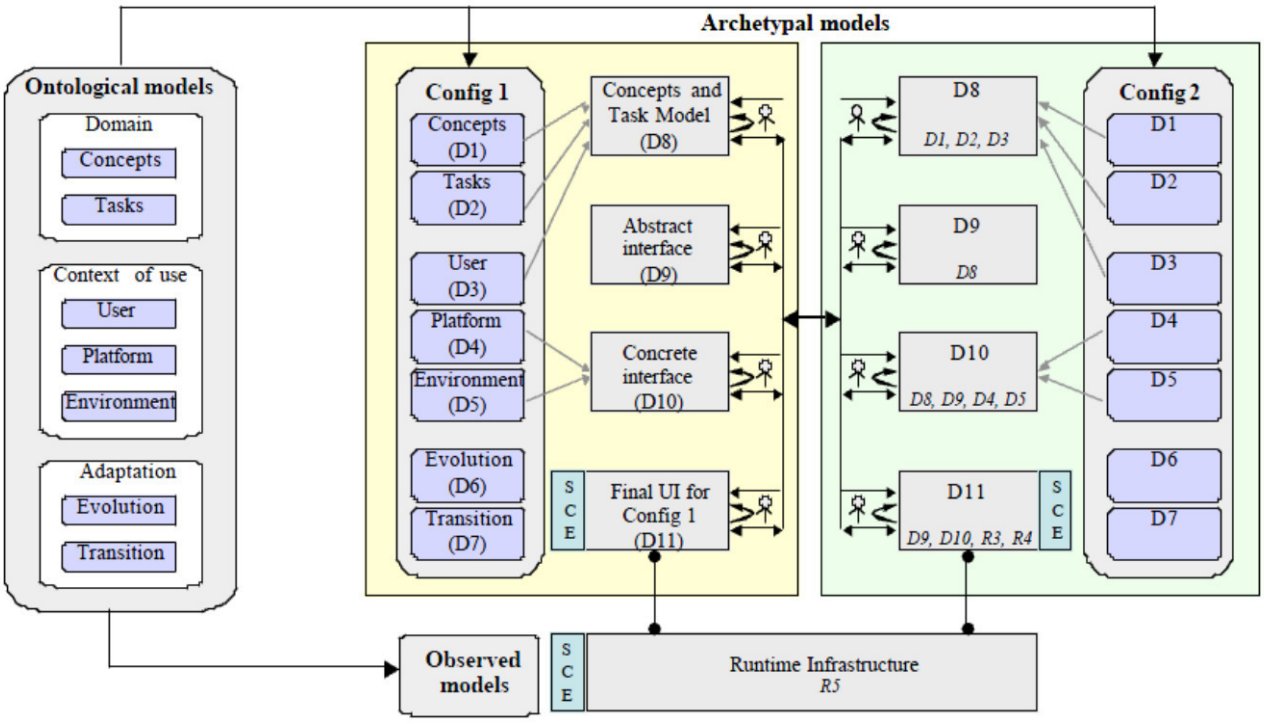

Fig. 1 CAMELEON Reference Framework

understanding a model-based UI development process. Model-based approaches [7], which rely on high-level specifications, provide the foundations for code generation and code abstraction. The framework fuses together different models that influence the overall UI adaptation. As shown in Figure 1, the framework covers the inference process from high-level abstract descriptions to run-time code, using a four-step reification process: from Concepts-and-Tasks Model (CTM), to Abstract User Interface (AUI), to Concrete User Interface (CUI), to Final User Interface (FUI). The CTM brings together concepts and tasks descriptions produced by designers for a particular interactive system and a particular target. The AUI is a universal description of the domain concepts and functions in a way that is independent of the UI implementation (in terms of UI widgets). At the CUI level the look and feel of a UI is defined, but the description is still device-independent. Finally, the FUI is expressed in a format suitable for a specific end-device and is tailored for this device. At each step the reification is influenced by the "context of use", defined as a set of parameters describing a user, a platform and the environment. Most of this process belongs to the design-time phase. The run-time phase includes the last reification from the CUI (device-independent) to the FUI (device-specific) and translations between FUI variants.

Both UI adaptation phases are different in nature. In our research on device-independent systems we do not address general UI design issues and we focus on the run-time UI adaptation phase, assuming that the design-time phase produces a device-independent UI description, which is used as a starting point for the run-time UI adaptation.

III. Device-Independent Architecture

The Device-Independent Architecture (DIA) has been proposed to facilitate analysis and development of applications that can be made available to users via any capable device from the large, diverse and fast growing pool of Internet-enabled end-devices – i.e., devices that are used directly by users to interact with an application, but not sensors that passively record a state of an environment. As presented in [8], the idea of DIA originates from the Service-Oriented Architecture, where systems are decomposed into atomic services, and processes use such services without knowing much about their implementation. A similar approach can be used to decompose an end-device. Each end-device, be it a laptop or a smartphone, provides: resources, services, and user interaction channels. Resources encompass processing power, memory and storage. Services are providers of context information, such as location, temperature, light intensity, and data from other types of sensors. User interaction channels (both incoming and outgoing) are the means to communicate with a user and include: screen, vibration, keyboard, microphone, camera, etc. The key concept is to use external resources, instead of what is provided by an end-device, and to generalize the way services and user interaction channels are accessed. Therefore, in DIA, the separation of application from end-

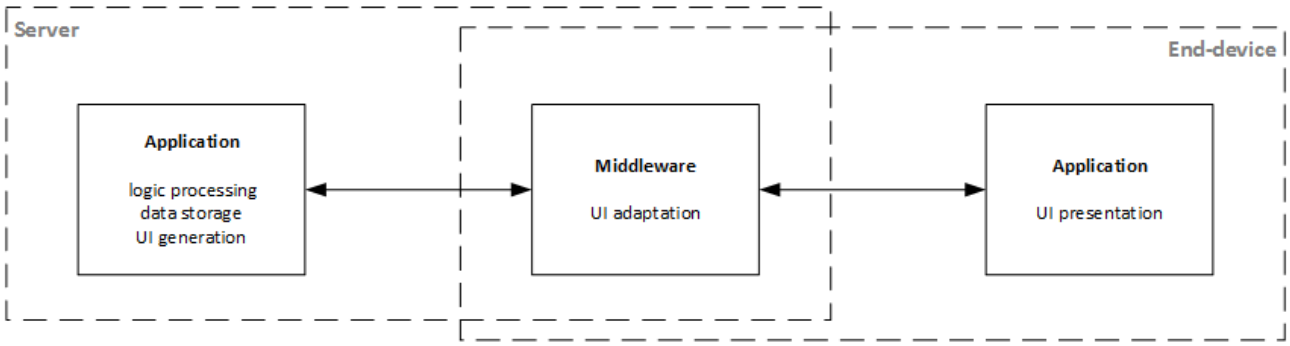

Fig. 2 Device-Independent Architecture diagram

devices, which enables the device independence, is achieved by:

- executing an application outside of end-devices,
- accessing sensor data provided by a device via a standardized API,
- using universal UI descriptions, and
- communicating with a user via a set of well-defined user interaction channels.

The execution of the application on external resources ensures that the application logic does not depend on the hardware or software platform of an end-device. The interesting consequence is that, in this architecture, end-devices could be deprived of their general purpose resources, as these resources are not needed. Services publish data in service-specific formats (e.g., location coordinates for a geolocation service, numerical data for a temperature sensor, and so on) independently of their implementation on a particular end-device. Therefore, it is feasible to build a middleware providing a device-independent API, such as the one proposed in Wolfram Language [9], to access such services. The usage of a universal UI description is a key requirement for making the UI of an application independent of parameters of user communication channels available on a given end-device (e.g. screen size and pixel density).

However, to enable a UI presentation tailored to parameters of a specific end-device, the generic UI description has to be properly adapted before reaching the user. That is why we have decided to research whether DIA-based software may use existing real-time UI adaptation methods.

IV. Model and Analysis

Run-time UI adaptation is a process that transforms a high-level, device-independent UI description (often model-based) prepared at design-time into a final UI presentation. In ideal situations, the high-level UI description may be presented in different ways depending on the UI modality of available user communication channels. For example, presentation of the same UI could be done on screen (Graphical User Interface (GUI)) or via speakers (e.g. Voice User Interface (VUI)). In general, the execution of a run-time UI adaptation process requires three parameters: the UI description, the content and a context of use. The content is used to fill-in the UI. The context of use influences the UI adaptation process and allows tailoring the final UI to the user, her end-devices and situation (location, time, etc.).

A. Run-time UI adaptation model

To be able to analyze the UI adaptation in different application architectures we have defined a simplified model of the UI adaptation and generation process. We call it the GARP model. The GARP model, presented in Figure 3, is composed of four main steps:

Step 1: input gathering (G). At the beginning of the process it is necessary to gather all input required for UI adaptation. The result of this step is a triplet of: UI

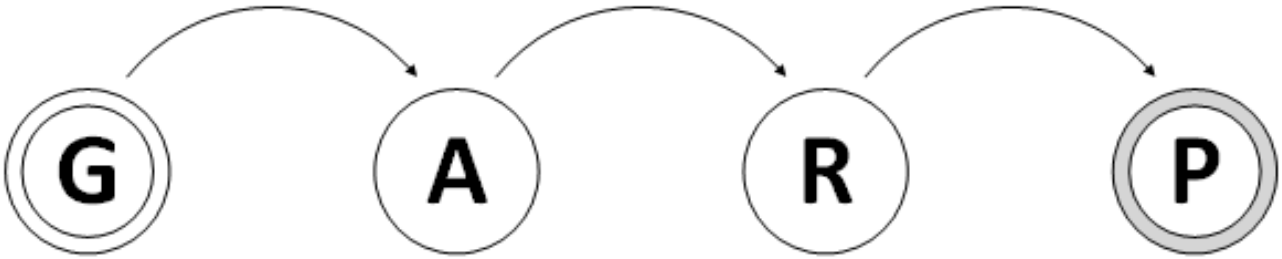

Fig. 3 Model of the UI adaptation and generation process

description, content and context.

Step 2: adaptation (A). In this step the content is used to fill-in the UI and the context is used to guide the transformation of the UI description into a final UI tailored for the user, her end-devices and situation. The result of this step is a device-specific UI description encoded with a specialized UI language such as HTML, QML, etc.

Step 3: rendering (R). The device-specific UI description provided by step 2 is interpreted here, in step 3 and the final UI presentation form is calculated. The final UI presentation form is data prepared for a specific user communication channel, e.g. pixels for screen or audio bits for speakers.

Step 4: presentation (P). The last step of the process is about presenting the UI to the user using a specific user communication channel of a specific end-device, e.g. showing images on screen or playing audio through speakers.

To make the analysis easier to follow, the model represents only the way towards a user and ignores the process of recording and interpreting user actions. Nevertheless, the path from a user can be modelled in a similar way, so our claims are valid for the whole user interaction loop.

For the analysis we have identified three classes of systems:

Client-side adaptation systems (CSA). Systems of this class include applications that are executed entirely on an end-device (a.k.a. local applications) and client-server applications with UI adaptation done on the client side.

Server-side adaptation systems (SSA). This class includes client-server applications with UI adaptation performed on the server side.

DIA-based systems (DIA), which include applications based on the Device-Independent Architecture.

Our goal is to see how the UI adaptation process differs among these classes of systems and how these differences influence the applicability of known UI adaptation methods. We acknowledge existence of in-between solutions. However, these three classes were selected to clearly show differences in the UI adaptation process.

B. UI adaptation in CSA systems

The UI adaptation and generation process in CSA systems is done either entirely on an end-device (client side) or the G step is supported by the server side, which provides for example the content, UI description or user preferences. However, the fragment of the context gathering related to the end-device is local, so the G step can be seen as a task performed jointly by the server side and the client side. The way the G step is performed (locally or split between server and client sides) does not influence the actual UI adaptation, because from the point of view of the A step the results of the G step are always provided in the same way – locally.

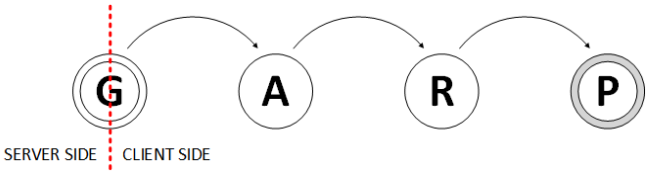

Fig. 4 GARP model in CSA systems

In CSA systems the A step may be implemented using existing UI adaptation methods designed for the use on an end-device. These methods are optimized for potentially limited processing capabilities of end-devices and are closely related to end-device characteristics and usage scenarios.

The local UI adaptation methods include solutions built-in into iOS and Android mobile operating systems and used by multiple mobile applications that run on various smartphones and tablets. On these mobile platforms the main issue is the diversity of screen sizes and pixel densities, so it is assumed that each application provides multiple variants of graphical assets (tailored to different screen densities) and some kind of a flexible layout that can be recalculated for any screen size. The drawback of these UI adaptation methods is that they are designed to cope with hardware parameters of a 'standard device' (in most cases a device with a touchscreen). Any UI adaptation that is supposed to take into account for example user preferences or non-standard devices, is not supported by the platform and has to be implemented manually.

The use of the server side for the G step usually does not change the fact that the adaptation implemented on the client side is somehow bound to the characteristic of an end-device. In our previous research [10] - [12] we have analyzed solutions that go beyond this local-only approach and use the server side to provide UI adaptation hints embedded in the high-level UI description provided by the G step, but even such extensions do not change the fact that the UI adaptation itself is device-specific, which makes it hard to reuse on other types of devices (devices with different hardware components, e.g. with two screens).

C. UI adaptation in SSA systems

In SSA systems the two initial steps: G and A, are performed on the server side, and the two other steps: R and P, are performed on the client side. The server gathers all input data, runs the UI adaptation and sends the device-specific UI description to the client. The client then interprets the UI description and presents it to a user.

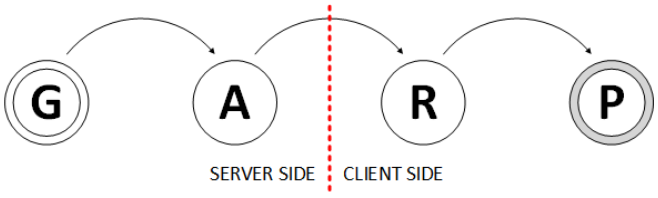

Fig. 5 GARP model in SSA systems

The SSA systems approach the issue of portability of the UI adaptation, shown for CSA systems, by implementing the A step on the server side. Such approach means that the UI adaptation is not bound by the performance of an end-device and can use external services to support the UI adaptation task (e.g. multimedia converters). Results of our previous research on UI adaptation in SSA systems [12] - [15] confirm that the A step in SSA systems may accommodate end-devices with disparate hardware configurations by using multiple or dynamic UI adaptation scenarios. However, the result of the A step is still interpreted on an end-device. Therefore is susceptible to differences in the final rendering and presentation on different end-devices. So full control of the resulting UI is not possible.

D. UI adaptation in DIA systems

The Device-Independent Architecture is based on an assumption that the whole processing is done outside of an end-device (the client side) and the end-device receives a pre-rendered UI ready for presentation, without the need for any interpretation. So in the case of DIA systems all three initial steps of the GARP model are done on the server side and only the P step is performed on the client side. The data transferred between the server and the client is usually a stream (e.g. a video or audio stream) or a static UI state (e.g. an image of the UI to be presented on screen or audio file to be played through speakers) ready to be presented on an end-device.

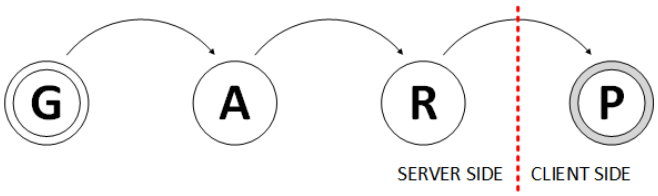

Fig. 6 GARP model in DIA systems

The DIA approach enables full control over the final UI presented to a user by implementing also the R step on the server side. UI adaptation methods used in DIA systems can be still the same as for SSA systems, but the fact that the end-device handles only the P step ensures that devices will not show a UI in a way that deviates from the designer intentions. The consequence of moving the R step to the server side is a different kind of data being transferred between the server side and the client side. In SSA systems, the client side receives a device-specific UI description encoded in a specialized UI language. In DIA systems, the server side has to send either a continuous stream of data tailored for specific user communication channels (e.g. video stream for a screen or an audio stream for speakers) or a static UI state composed of multiple files that are targeted at different user communication channels (e.g. image files to be shown on a screen or audio files to be played through speakers). The main difference here is the increased size of data that has to be transferred. More data to transfer could mean longer response times, but our previous research [16] showed that in analyzed scenarios DIA-based systems can still maintain proper response times to UI interactions initiated by a user, despite the increased size of transferred data.

V. CONCLUSION

The Device-Independent Architecture can be treated as a special case of a client-server architecture, in which the client side is assumed to be an extremely thin client and in which all the processing is done on the server side. The DIA takes it even further and defines the client side as a set of user communication channels, which makes it possible to model multiple end-devices as a complex client device, but this distinction does not necessarily change the way the UI adaptation is performed. Therefore, DIA systems may use the same existing UI adaptation methods that were designed for SSA systems, or for client-server systems in general.

The main difference is related to the fact that in DIA systems the data transferred between the server side and the client side tends to be larger than in the case of SSA systems. Therefore, network usage optimization is crucial. Especially that the transmission delay will directly influence the UI responsiveness. Moreover, used communication protocols and formats of presentation data sent to an end-device have to be negotiated beforehand, to make sure that the end-device is able to receive and present it properly.

Summarizing, despite using a different implementation of the GARP model, the DIA systems may use existing real-time UI adaptation methods. The difference in the implementation of GARP model influences only the optimization of the UI adaptation and generation process. In CSA systems the key optimization aspect is end-device performance. In SSA systems the key optimization aspect is uniform interpretation of the device-specific UI description. While, in DIA systems the key optimization aspects are network-related. First, it is necessary to use data formats that minimize the amount of bits that have to be transmitted. Second, it is crucial to use the best possible data transfer protocols. The best are the ones with low overhead, low latency and support for QoS. Both points should be taken into account by the run-time UI adaptation task, because the nature of a UI (state-based or continuous) may influence the set of suitable transmission protocols.

We expect that different protocols will be best suited for different user interaction scenarios. Our next research goal in this area is be to identify user interaction patterns and UI design patterns, which could be used to define rules for selecting the best protocol and data formats for a given user interaction scenario.

REFERENCES

[1] Chmielewski, J., Towards an Architecture for Future Internet Applications, in: The Future Internet, vol. Lecture Notes in Computer Science 7858 , Springer Berlin Heidelberg, 2013, pp. 214-219, ISBN 978-3-642-38081-5, DOI 10.1007/978-3-642-38082-2_18

[2] Meixner, G., Calvary, G., Coutaz, J., Introduction to Model-Based User Interfaces, W3C Working Group Note, December 2013, http://www.w3.org/2011/mbui/drafts/mbui-intro/

[3] Sottet, J. S., Calvary, G., Favre, J. M., & Coutaz, J., Megamodeling and metamodel-driven engineering for plastic user interfaces: MEGA-UI. In Human-Centered Software Engineering, pp. 173-200. Springer London. 2009, DOI 10.1007/978-1-84800-907-3_8

[4] Jaouadi, I., Ben Djemaa, R., & Ben Abdallah, H., Interactive Systems Adaptation Approaches: A survey. In ACHI 2014, The Seventh International Conference on Advances in Computer-Human Interactions, pp. 127-131. March 2014, ISSN: 2308-4138, ISBN: 978-1-61208-325-4

[5] Ye, J. H., & Herbert, J., Framework for user interface adaptation. In User-Centered Interaction Paradigms for Universal Access in the Information Society, pp. 167-174. Springer Berlin Heidelberg. 2004 DOI 10.1007/978-3-540-30111-0_14

[6] Calvary, G., Coutaz, J., Bouillon, L., Florins, M., Limbourg, Q., Marucci, L., Paternò, F., Santoro, C., Souchon, N., Thevenin, D., Vanderdonckt, J., The CAMELEON Reference Framework, Deliverable 1.1, CAMELEON Project, 2000

[7] Szekely, P., Retrospective and challenges for model-based interface development, in Design, Specification and Verification of Interactive Systems '96, Eurographics 1996, pp. 1-27. Springer Vienna. 1996 DOI 10.1007/978-3-7091-7491-3_1

[8] Chmielewski, J., and K. Walczak, Application Architectures for Smart Multi-device Applications, in: Proceedings of the Workshop on Multi-device App Middleware 2012, Workshop on Multi-device App Middleware 2012, Montreal (Canada), December 3 – 7, 2012, ACM, New York, 2012, pp. 5:1 - 5:5, ISBN 978-1-4503-1617-0, DOI 10.1145/2405172.2405177

[9] Wolfram Language for Knowledge-Based Programming, https://www.wolfram.com/language/, 2015

[10] Chmielewski, J., K. Walczak, and W. Wiza, Mobile Interfaces for Building Control Surveyors, in: Software Services for e-World, IFIP Advances in Information and Communication Technology, Vol. 341, ed. Cellary, W., and E. Estevez, The 10th IFIP WG.6.11 Conference on e-Business, e-Services and e-Society I3E 2010, Buenos Aires, Argentina, November 3-5, 2010, Springer, 2010, pp. 29-39, ISBN 978-3-642-16282-4 DOI 10.1007/978-3-642-16283-1_7

[11] Rykowski, J., and J. Chmielewski, Automatyczna generacja zintegrowanego interfejsu człowiek-maszyna na potrzeby inteligentnego budynku , in: Inteligentne budynki - teoria i praktyka, ed. Mikulik, J. , Oficyna Wydawnicza Text, Kraków, 2010, pp. 166-188, ISBN 978-83-60560-54-9

[12] Chmielewski, J., K. Walczak, W. Wiza, and A. Wójtowicz, Adaptable User Interfaces for SOA Applications in the Construction Sector, in: SOA Infrastructure Tools - Concepts and Methods, ed. Ambroszkiewicz, S., J. Brzeziński, W. Cellary, A. Grzech, and K. Zieliński, Wydawnictwa Uniwersytetu Ekonomicznego w Poznaniu, Poznań, 2010, pp. 493-469, ISBN 978-83-7417-544-9

[13] Jansen, A., Bronmark, J., & Chmielewski, J. (2013). Method of adapting a user interface in industrial process monitoring and control applications. The Swedish Patent and Registration Office. SE 1300702-6

[14] Walczak, K., W. Wiza, D. Rumiński, J. Chmielewski, and A. Wójtowicz, Adaptable User Interfaces for Web 2.0 Educational Resources, in: IT Tools in Management and Education - Selected Problems, ed. Kiełtyka, L., Wydawnictwo Politechniki Częstochowskiej, Częstochowa, 2011, pp. 104-124, ISBN 978-83-7193-508-4

[15] Walczak, K., J. Chmielewski, W. Wiza, D. Rumiński, and G. Skibiński, Adaptable Mobile User Interfaces for e-Learning Repositories, IADIS International Conference on Mobile Learning, Avila (Spain), Marc 10-12, 2011, IADIS, 2011, pp. 52-60, ISBN 978-972-8939-45-8

[16] Chmielewski, J., Device-Independent Architecture for Ubiquitous Applications, in: Personal and Ubiquitous Computing, Volume 18, Issue 2, pp 481-488, Springer London, 2014, DOI 10.1007/s00779-013-0666-y