Guidelines for integrating personas into software engineering tools.

FAILY, S. and LYLE, J.

2013

Guidelines for Integrating Personas into Software Engineering Tools

Shamal Faily
Department of Computer Science
University of Oxford, Oxford OX1 3QD, UK
firstname.lastname@cs.ox.ac.uk

John Lyle
Department of Computer Science
University of Oxford, Oxford OX1 3QD, UK
firstname.lastname@cs.ox.ac.uk

ABSTRACT
Personas have attracted the interest of many in the usability and software engineering communities. To date, however, there has been little work illustrating how personas can be integrated into software tools to support these engineering activities. This paper presents four guidelines that software engineering tools should incorporate to support the design and evolution of personas. These guidelines are grounded in our experiences modifying the open-source CAIRIS Requirements Management tool to support design and development activities for the EU FP7 *webinos* project.

Author Keywords
Personas; Design Rationale; CAIRIS; CAQDAS; XML

ACM Classification Keywords
H.5.2 User Interfaces: User-centered design

General Terms
Human Factors; Design.

INTRODUCTION
Personas – behavioural specifications that embody the goals and needs of archetypical users – have been a popular technique in user-centered design practice since they were first introduced over 10 years ago [4]. Personas were introduced to deal with developer biases arising from the word *user*. Such biases lead programmers to bend and stretch assumptions about what users should do. By building systems with only specific users in mind, developers only have to focus on those requirements necessary to keep these users happy. The idiosyncratic detail associated with personas also makes them communicative to a variety of different technical and non-technical stakeholders.

In recent years, interest in personas and the personification of archetypical users has extended to the software engineering communities. For example, Constantine [3] describes how models of user roles, which are analogous to personas, can drive visual interface design, while Roberts [17] claims that personas can supplement UML models by contextualising descriptions of user roles. Surprisingly, however, despite several proposals for integrating personas methodologically with software engineering [1, 2, 15], there has been little work describing how software engineering tools should be augmented to support their creation, usage, and on-going maintenance. We believe this is unfortunate. From a user-centered design perspective, it may be acceptable to create and use personas without the use of software. However, from an engineering perspective, personas may be intricately woven into the traceability of different models, and justify design decisions at many levels of abstraction. Failure to properly specify, validate, or maintain personas may lead to precisely the sort of problems they were originally designed to address. Consequently, while personas are not rigorously specified, they are still models that contribute to the engineering of interactive systems.

It seems reasonable that personas be accorded the same level of tool-support as other software design models. However, to date, the focus of software engineering tools has been to support the design and development of software, not user-centered design artifacts like personas. This means we need to understand how personas can be integrated into software tools to ensure they help, rather than hinder, software engineering practice.

In this paper, we present four guidelines that enable software engineering tools to support the creation, use, and on-going evolution of personas. After describing the related work that motivated our work, we briefly introduce the open-source CAIRIS software tool, and the EU FP 7 *webinos* project that formed our context of study. We then present four guidelines based on our experiences using personas to support the design and development of the *webinos* platform. We conclude by discussing some of the consequences of our work.

RELATED WORK
Despite their value in supporting the design and development process, there is a surprising lack of both literature and software tools for supporting the creation, analysis, and on-going management of personas. One of the few current examples of software for persona creation and management is *Persona* [13] from Mariner Software. This tool manages information about personas, and predicts their behaviour. However, the tool itself is designed to support creative writing so some of its capabilities, such as re-using hero and villain archetypes, are of limited use to software design. Moreover, while the

tool's capabilities for exploring persona relationships may have some value when designing collaborative systems, engineers may prefer to focus on explicit aspects of these interactions using more formal, UML based models.

The software engineering community have long argued that personas can be used to bridge usability and software engineering processes. For example, recent work by Schneidewind et al. [18] argued that personas can support all stages of the requirements engineering process, from requirements elicitation with use cases through to requirements validation. This is made possible because the purposes of personas, such as modelling and prioritising, align with different requirements engineering activities that require insights about users. However, while personas help support these activities, their effect may be marred without effective tool-support. For example, Schneidewind et al. claim that personas help prioritise requirements because they help understand the requirements of use case actors. However, the needs of personas may conflict, and such conflicts may not be immediately obvious from descriptions and use cases alone. It is, therefore, necessary to view the different relationships personas have with different design models, and ensure both personas and associated models are shared between team members and other stakeholders.

One of the few examples of software that integrates personas with different software design models is CAIRIS (Computer Aided Integration of Requirements and Information Security) [6]. CAIRIS is an open-source Requirements Management tool that was initially designed to support the specification of interactive secure systems. In addition to managing requirements models such as use cases and goal models, CAIRIS also manages the data associated with several security and usability engineering models, such as risks, scenarios, and personas. CAIRIS complements the use of specific security, requirements, and usability engineering techniques. If these techniques were properly applied, then models arising from them could be directly entered or imported into CAIRIS. The implications of personas in a system's design would then be amenable to some automated analysis. For example, [7] illustrates how the alignment between personas, scenarios, and risk analysis models can be used to visualise both the security and usability impact of different scenarios to personas.

As a product of 'autobiographical' design research [16], CAIRIS had previously been used in several projects where a single designer was responsible for eliciting all of the data managed by the tool. However, on larger projects, designers collecting data may not be the same people responsible for building or using personas. In such situations, designers may be uncomfortable using personas if the data upon which they are based is unavailable. For example, in a recent study examining how designers use personas, Matthews et al. [14] found that many designers find personas too abstract, and instead prefer using the data upon which these are based. Such data allows designers to make their own sense of user data. This suggests that if software tools are to adequately support personas, they also need to support the qualitative data analysis processes that create them. However, to date, Computer Aided Qualitative Data Analysis (CAQDAS) software tools, techniques, and general insights for carrying out qualitative data analysis, e.g. [11] remain largely restricted to academic researchers rather than software engineers.

INTEGRATING PERSONAS INTO WEBINOS

To understand how personas can be integrated into software tools, we customised CAIRIS for the EU FP 7 *webinos* project [22]. This project integrated the use of personas with its design and development activities.

webinos is a software infrastructure (*webinos*) for running web applications across different device platforms [10]. The project team was drawn from 24 organisations across Europe, including universities, network providers, handset and car manufacturers, mobile software houses and market analysts.

To understand the expectations of prospective *webinos* users and application developers, ten personas was created. These personas were created by a team of five designers, drawn from across the *webinos* consortium; this team was responsible for not only creating the personas, but also maintaining them throughout the life of the project. To create the personas, the designers elicited factoids about prospective users from a variety of data sources. Affinity diagramming was used to categorise these factoids into clusters of potential behaviour. These clusters were structured using argumentation models [19] to motivate individual persona characteristics. More information about process used to create these personas, and the personas themselves, can be found in [20]. Once the personas had been created, the designers entered the persona argumentation structures and narratives into CAIRIS. These designers were also responsible for maintaining the personas throughout the project.

In addition to personas, scenarios were also created by project team members to envisage how *webinos* might be used, and describe some of the unintended consequences that might arise as a result. The design team incorporated and, where necessary, revised these scenarios based on the different personas. These scenarios were then entered into CAIRIS, together with related use cases and requirements for different functional areas of *webinos* [21].

As the project progressed, personas were used to motivate more detailed design and implementation activities. Personas played an important role in supporting an architectural risk analysis of the *webinos* software platform [9], and were used by developers to motivate new feature requests, e.g. [12]. When necessary, new personas were also created. For example, one persona was created for a 5 year old child to understand how he might interact with a *webinos*-enabled travel game for young children.

GUIDELINES

Based on our experiences customising CAIRIS for *webinos*, we have elicited the following four guidelines for developers wishing to integrate personas into their own software engineering tools.

- *Make persona characteristics explicit*
  Software Engineering tools should make the rationale for

persona characteristics available from the interfaces where persona narratives are displayed.

- *Integrate qualitative data analysis*
  Software Engineering tools should provide the analytical support necessary to create and maintain personas.

- *Facilitate persona interchange*
  Software Engineering tools should serialise personas in a format that encourages interchange and maintenance by developers that may be using different tools.

- *Revision control personas*
  Software Engineering tools should facilitate the revision control of personas.

We motivate and demonstrate these guidelines in the following sub-sections.

Make persona characteristics explicit

Personas were a new concept to most people on the project, with some developers being hesitant to trust what they considered to be fictional narratives. This meant that the analysis underpinning the personas needed to provide rationale for their descriptions, at the point when this rationale might be needed.

We believed that aligning the narrative of persona characteristics with their argumentation models would provide this missing rationale. However, questions about these characteristics and their validity still arose when interacting with personas using CAIRIS. To address this, we decided to link context-specific argumentation models within the persona dialog controls. Each persona dialog contained folders for the persona's activities, attitudes, aptitudes, motivations, and skills; these were based on behavioural variable types suggested by [5]. We added a context-sensitive menu to each folder to allow the visualisation of all the characteristic argumentation models associated each behavioural variable type.

Figure 1 illustrates the visualisation of the argument underpinning the persona characteristic 'Contextual variety encourages rather than discourages user-centeredness'. Based on Toulmin's model of argumentation, the white boxes act as the grounds for the characteristic's claim, while the blue boxes act as warrants connecting the grounds to the characteristic. The link to the data source document is indicated by the grey boxes linking the grounds and warrant objects. Finally, the dotted box shows a qualifying term indicating how confident the analyst is about this characteristic.

Applying this guideline helped the team identify personas that were overly general or ambiguous. In a number of cases, we identified personas that were built incorrectly by developers and analysts, who wrote the person narratives before following the argumentation process; this effectively bypassed the prescribed processes for creating personas. The argumentation model visualisation was useful in spotting and correcting such personas by drawing attention to the fallacies underpinning the personas' characteristics and, together with the personas' designers, walking through how the prescribed process — and the argumentation models in particular — identified these fallacies, while also eliciting new characteristics about the persona.

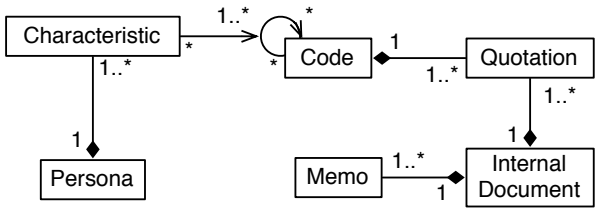

Figure 2. Meta-model of a persona's qualitative data analysis elements

Integrate qualitative data analysis

The argumentation models provided some assurance about a persona's characteristics, but useful insights which arose while creating the personas were lost once the personas were created. To capture such insights, it is not enough for tools for interact with persona data - they need to support the analytical processes necessary to create and maintain them as well.

The Persona Case framework set out in [8] provided an approach for qualitatively analysing this data and, based on this analysis, developing new persona characteristics. This framework, however, assumed that a dedicated CAQDAS tool was available to carry out this analysis. Unfortunately, financial constraints meant that the potential commercial tools could not be used, and no suitable open-source alternative was available. Consequently, we decided to update CAIRIS to incorporate all the elements necessary to support qualitative data analysis while conforming to the framework described by [8].

To examine the efficacy of this integration, we used CAIRIS and the Persona Case framework to augment an existing persona with new characteristics. This persona (Jimmy) was based on application developer, which was a key audience for *webinos*. The empirical data upon which Jimmy's characteristics were based were collected was selected during three focus group sessions organised for potential *webinos* application developers. Focus group participants were recruited based on characteristics they shared with Jimmy.

The class diagram in Figure 2 shows the concepts incorporated into CAIRIS' data model and interfaces. The focus group reports were imported into CAIRIS as *internal document*s. The reports were qualitative coded, and text segments were annotated as *quotations* that were categorised according to a specific categorical *code*. Alternatively, where text segments raised particular questions or insights then these could be annotated with *memo*s. Conceptual relationships between codes could be modelled, where each relationship corresponded to a persona *characteristic*. The Persona Case framework requires the quotations associated with each code to be both further categorised and labelled according to the role it plays in each characteristic's argumentation model.

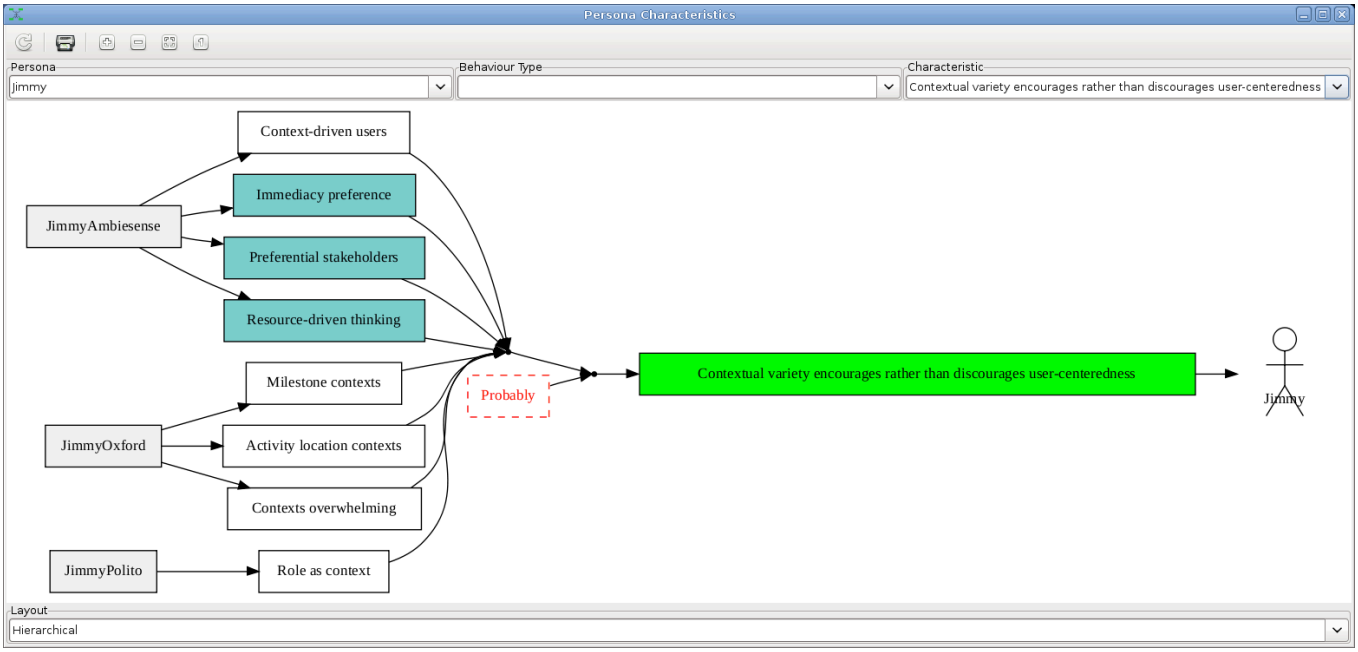

Figure 1. Visual model of a persona characteristic's grounding

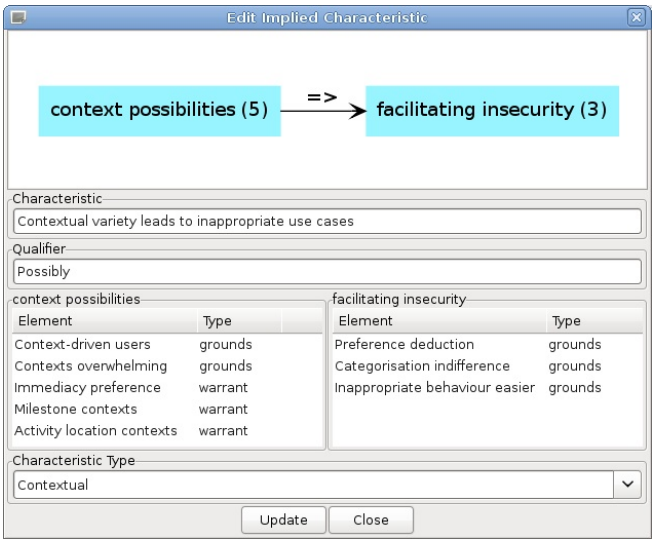

Figure 3. Details of qualitative grounding of a persona characteristic

To support the specification of this information, interfaces were added not only for adding and managing codes, quotations, and code relationships but also, as Figure 3 shows, the role each quotation played in a characteristic's argumentation model.

Facilitate persona interchange

Making CAIRIS available to everyone on the project made model interchangeability a concern. CAIRIS facilitated model interchange using XML; models would be exported to XML on one running instance of CAIRIS, and imported on another. Due to the complexity of the underlying CAIRIS database, which contained over 320 tables, the *webinos* models were spread across multiple XML documents; each document was structured according to a Document Type Description (DTD). These DTDs structured elements according to different model categories. For example, requirements model elements were structured according to a 'goals' DTD, while usability model elements such as personas and scenarios were structured based on a 'usability DTD'. Because there were dependencies between different XML documents that could not be easily resolved, these needed to be imported into CAIRIS in a particular order to ensure referential integrity in the database was maintained.

Developers were comfortable using XML to exchange models, but not all developers wanted to use CAIRIS. CAIRIS, however, played an important role is analysing the impact of different models on persona characteristics and vice-versa. Therefore, to ensure that team members would maintain personas, model interchangeability had to be supported at the level of personas, rather than at the coarse grained levels of the DTD. To do this, we modified the DTD for CAIRIS 'usability' elements to make the exchange of personas easier. An excerpt of the revised usability DTD is visualised in Figure 4. Class names correspond with element names, and the left-right order in the diagram indicates the order of elements within the DTD. Our aim in modifying this DTD was to allow personas and all their supplemental data to be externalised in a single XML document. We also wanted to make the structure of these documents as close as possible to CAIRIS' conceptual model of personas. This meant that, while the DTD was sub-optimal when compared to XML schema validation, it was easier for most team members to understand and maintain.

We believe this approach to persona interchange was successful because, while CAIRIS was only consistently used by a

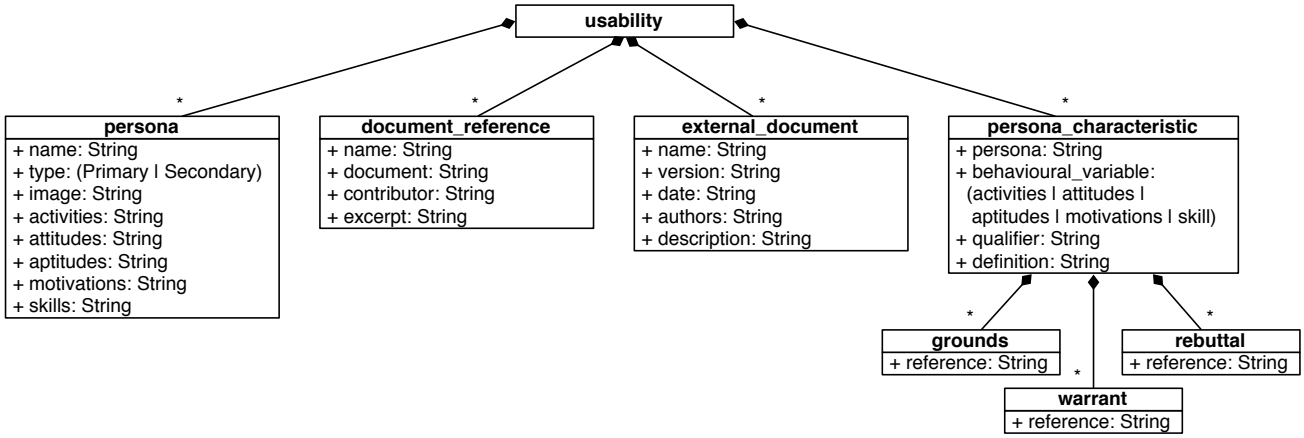

Figure 4. Class diagram of persona DTD structure

few team members, the personas still continued to be maintained and used. This is because scripts were created to export the CAIRIS models to the project wiki for easy viewing. If changes were necessary, the XML models could be edited directly using simple text editors.

Revision control personas

The interchangeability of personas raised questions about how they should be treated with respect to other design models. We were concerned not only with the traceability of personas with other models, but also how these would be maintained as *webinos*' design evolved. This was particularly important when personas were used to discuss implementation specific features, e.g. [12]. Consequently, if personas were to be accorded the same consideration as other models by the project team they would need to be placed under revision control. This is because the other design models were already under revision control and made openly available as a GitHub project [23].

We implemented this guideline in two ways. First, as described in the previous section, each persona and its associated qualitative data was stored in a single XML document. However, because some persona argumentation elements were used to ground multiple personas, importing individual persona XML files would lead to the import of duplicate data, thereby breaking the CAIRIS database's entity integrity. Therefore, the import logic was modified to ensure such duplicates would not be added. Second, the persona files were stored in a single directory on GitHub, and the model *build* script was revised to import each individual persona into the CAIRIS database. This meant personas would be kept in one place for ease of access, and any syntactical errors in the XML documents, or referential integrity errors resulting from problematic traceability links, could be addressed while models were being imported into CAIRIS.

CONCLUSION

In this paper, we illustrated how personas could be better supported with software engineering tools, and presented four guidelines based on our experiences modifying CAIRIS for the *webinos* project. As a result, we believe this paper makes three important contributions.

First, we have drawn attention to the need for tool-support for personas, and discussed the consequences of this need remaining unaddressed. We hope others in the community share our concerns and will be inspired to join us by continuing work in this area.

Second, we have provided practical examples of how software tools can be augmented to better support personas when engineering interactive systems. The need for supporting persona interchangeability and revision control may seem self-evident, but our work has demonstrated how such guidelines can be implemented. We are conscious of the limitations of these guidelines, which are grounded only in the experiences in a single project. However, as a large-scale project, we believe that these experiences can be generalised.

Third, we believe these guidelines raise broader methodological questions. For example, our approach for revision controlling personas was appropriate for *webinos* as all source data had been anonymised. However, such an approach may not be appropriate when working with sensitive data that needs to be anonymised or access controlled. While such concerns are orthogonal to this paper, provisioning CAIRIS for the creation and maintenance of personas highlighted issues that might otherwise have remained unaddressed.

ACKNOWLEDGMENTS

The research described in this paper was funded by EPSRC *EUSTACE* project (R24401/GA001), and EU FP7 *webinos* project (FP7-ICT-2009-05 Objective 1.2).

REFERENCES

1. Behrenbruch, K., Atzmüller, M., Evers, C., Schmidt, L., Stumme, G., and Geihs, K. A personality based design approach using subgroup discovery. In *Proceedings of the 4th international conference on Human-Centered Software Engineering*, HCSE'12, Springer-Verlag (Berlin, Heidelberg, 2012), 259–266.
2. Castro, J. W., Acuna, S. T., and Juristo, N. Integrating the personas technique into the requirements analysis

activity. In *Proceedings of the 2008 Mexican International Conference on Computer Science*, IEEE Computer Society (2008), 104–112.

3. Constantine, L. Users, Roles, and Personas. In *The persona lifecycle: keeping people in mind throughout product design*, J. Pruitt and T. Adlin, Eds. Morgan Kaufmann, 2006, ch. 8, 498–519.
4. Cooper, A. *The Inmates Are Running the Asylum: Why High Tech Products Drive Us Crazy and How to Restore the Sanity (2nd Edition)*. Pearson Higher Education, 1999.
5. Cooper, A., Reimann, R., and Cronin, D. *About Face 3: The Essentials of Interaction Design*. John Wiley & Sons, 2007.
6. Faily, S. CAIRIS web site. **http://github.com/failys/CAIRIS**, March 2013.
7. Faily, S., and Fléchais, I. Towards tool-support for Usable Secure Requirements Engineering with CAIRIS. *International Journal of Secure Software Engineering 1*, 3 (July-September 2010), 56–70.
8. Faily, S., and Fléchais, I. Persona cases: a technique for grounding personas. In *Proceedings of the 29th international conference on Human factors in computing systems*, ACM (2011), 2267–2270.
9. Faily, S., Lyle, J., Namiluko, C., Atzeni, A., and Cameroni, C. Model-driven architectural risk analysis using architectural and contextualised attack patterns. In *Proceedings of the Workshop on Model-Driven Security*, ACM (2012), 3:1–3:6.
10. Fuhrhop, C., Lyle, J., and Faily, S. The webinos project. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, ACM (New York, NY, USA, 2012), 259–262.
11. Lewins, A., and Silver, C. *Using software in qualitative research : a step-by-step guide*. SAGE, Los Angeles, 2007.
12. Lyle, J. JIRA Issue WP-596: Single authentication for peer-to-peer service sharing. **http://jira.webinos.org/browse/WP-596**, 2012 November.
13. Mariner Software. Persona. **http://www.marinersoftware.com/products/persona**, March 2013.
14. Matthews, T., Judge, T., and Whittaker, S. How do designers and user experience professionals actually perceive and use personas? In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, ACM (2012), 1219–1228.
15. Moundalexis, M., Deery, J., and Roberts, K. *Integrating human-computer interaction artifacts into system development*, vol. 5619 LNCS. Springer, 2009.
16. Neustaedter, C., and Sengers, P. Autobiographical design in hci research: designing and learning through use-it-yourself. In *Proceedings of the Designing Interactive Systems Conference*, DIS '12, ACM (2012), 514–523.
17. Roberts, D. Coping with complexity. In *Human-Centered Software Engineering: Integrating Usability in the Software Development Lifecycle*, A. Seffah, J. Gulliksen, and M. C. Desmarais, Eds. Springer, 2005, ch. 11, 201–217.
18. Schneidewind, L., Horold, S., Mayas, C., Kromker, H., Falke, S., and Pucklitsch, T. How personas support requirements engineering. In *Usability and Accessibility Focused Requirements Engineering (UsARE), 2012 First International Workshop on* (2012), 1–5.
19. Toulmin, S. *The uses of argument*, updated ed. Cambridge University Press, 2003.
20. webinos Consortium. User expectations on privacy and security. **http://webinos.org/content/webinos-User_Expectations_on_Security_and_Privacy_v1.pdf**, February 2011.
21. webinos Consortium. Updates on Requirements and available Solutions. **http://www.webinos.org/wp-content/uploads/2012/09/Updates_on_Requirements_and_available_Solutions_v1.1_public.pdf**, August 2012.
22. webinos Consortium. webinos web site. **http://webinos.org**, March 2012.
23. webinos Consortium. webinos design data repository. **https://github.com/webinos/webinos-design-data**, March 2013.