Chapter 9

Reporting a User Study on a Visual Editor to Compose Rules in Active Documents

Federico Cabitza
Università degli Studi di Milano-Bicocca, Italy

Iade Gesso
Università degli Studi di Milano-Bicocca, Italy

ABSTRACT

In the last years, researchers are exploring the feasibility of visual language editors in domain-specific domains where their alleged user-friendliness can be exploited to involve end-users in configuring their artifacts. In this chapter, the authors present an experimental user study conducted to validate the hypothesis that adopting a visual language could help prospective end-users of an electronic medical record define their own document-related local rules. This study allows them to claim that their visual rule editor based on the OpenBlocks framework can be used with no particular training as proficiently as with specific training, and it was found user-friendly by the user panel involved. Although the conclusions of this study cannot be broadly generalized, the findings are a preliminary contribution to show the importance of visual languages in domain-specific rule definition by end-users with no particular IT skills, like medical doctors are supposed to represent.

INTRODUCTION

During the past years we conducted a number of observational studies in the healthcare domain (Cabitza & Simone, 2012; Cabitza, Simone, & Sarini, 2009) to elicit requirements for the successful introduction of electronic document-based information systems in organizational settings that presented features of criticality and complexity that are typical of hospital settings, especially the feature of heavily relying on paper-based documents and hence typically exhibiting a strong resistance to change. In such settings, documents (i.e., forms, charts, reports, spreadsheets) provide end-users with the natural interface by which to interact with the underlying information system.

The main finding of these studies is that traditional electronic document management systems are often too rigid with respect to the local needs of end-users and to how the corresponding requirements can change over time. For this reason, unlike...
what happens in traditional software engineering approaches toward software adaptivity (e.g., see (Weyns, Malek, & Andersson, 2010) and (de Lemos et al., 2012) for an overview), which are aimed at providing software engineers with a set of tools that allow them to design and configure self-adaptive systems, we believe that end-users should be left being autonomous in making their document-based information systems “on their own” (Cabitza, Gesso, & Corna, 2011); this means that users should become as much as possible independent of IT providers (e.g., software specialists and application vendors) in tweaking and adapting their systems to their ever changing functional needs.

In this light, we conceived the WOAD framework (Cabitza & Gesso, 2011; Cabitza & Simone, 2010), which encompasses an End-User Development (Lieberman, Paternò, & Wulf, 2006) environment. In particular, WOAD is aimed at making end-users autonomous in their customization efforts in regard to two distinct aspects by means of two specific visual editors: the Template Editor (Cabitza et al., 2011) to build and maintain templates of digital documents; and the Mechanism Editor, to augment those documents with simple rules (i.e., mechanisms in WOAD) and make it active. In particular, mechanisms make documents able to react to the users’ interaction and to changes in the execution context in an asynchronous and proactive manner, and to modulate how the document content should “look like” (i.e., its layout affordances), in order to promote the end-users’ collaboration awareness (Gutwin & Greenberg, 1997) of what is going on and help them cope with the current situation. Mechanisms are simple if-then constructs that couple a set of document-related conditions with a set of actions that act on the documents’ structure and content. Despite their simplicity, defining rules can result to be a hard task for end-users: indeed, these latter ones are usually experts of their work domain and setting, but can have relatively low confidence with formalized languages and, in general, with programming concepts and the related constraints. In this light, our point is that visual languages are useful to fill in this gap and make end-users able to “program” the active behavior of their electronic artifacts in order to fulfill the requirement of autonomy we mentioned above.

In what follows, after a short survey of the main visual languages we considered to express rule-based mechanisms, we present the main aspects of our rule visual editor, and we describe an experimental user study that we conducted to evaluate the usability of the editor and, hence informally, the feasibility of the whole EUD-oriented approach. The research question that motivated this user study is whether a visual language can make “programming” an interface as easy as a child’s play, i.e., akin to using simple building blocks in order to accomplish simple tasks that do not require a specific training or a long acquired competence in programming.

BACKGROUND

Within the healthcare domain, i.e., the reference domain of our research activities, the need to make the definition of the user interfaces of an Electronic Patient Records (EPR) more flexible, typically more or less structured forms, so that these can be adjusted to better meet the local needs of each single group of practitioners has been recognized long ago and confirmed in a number of recent field studies (e.g., (Bringay, Barry, & Charlet, 2006; W. Chen & Akay, 2011; Mamlin et al., 2006; Morisson & Blackwell, 2009)). Nevertheless, despite this recognition, the necessary tailoring activities to this aim still require that IT professionals work together with the end-users of such systems, i.e., the clinicians, in order to be able to perform the due customizations in a timely and aptly manner. In other words, clinicians can not autonomously tailor their EPRs. Mamlin et al. (2006), for instance, proposed OpenMRS, an open-source, modular solution to allow the implementation of EPRs that