Science of Computer Programming 86 (2014) 89–102

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Evolutionary design of user interfaces for workflow information systems

Josefina Guerrero-García *

Faculty of Computer Sciences, Benemérita Universidad Autónoma de Puebla, Av. San Claudio y 14 Sur, Puebla, Mexico

HIGHLIGHTS

- We propose a *methodology* to support evolutionary user interface development.
- Relying on models simplifies the design of complex systems.
- Methodology facilitates business process automation.

ARTICLE INFO

Article history:
Received 20 July 2012
Received in revised form 16 April 2013
Accepted 1 July 2013
Available online 23 July 2013

Keywords:
User interfaces
UsiXML
Workflow
Information systems
Model driven engineering

ABSTRACT

In this paper we argue that user interface design should evolve from iterative to evolutionary in order to support the user interface development life cycle in a more flexible way. Evolutionary design consists of taking any input that informs to the lifecycle at any level of abstraction and its propagation through inferior and superior levels (vertical engineering) as well as the same level (horizontal engineering). This lifecycle is particularly appropriate when requirements are incomplete, partially unknown or to be discovered progressively. We exemplify this lifecycle by a methodology for developing user interfaces of workflow information systems. The methodology involves several models (i.e., task, process, workflow, domain, context of use) and steps. The methodology applies model-driven engineering to derive concrete user interfaces from a workflow model imported into a workflow management system in order to run the workflow. Instead of completing each model step by step, any model element is either derived from early requirements or collected in the appropriate model before being propagated in the subsequent steps. When more requirements are elicited, any new element is added at the appropriate level, consolidated with the already existing elements, and propagated to the subsequent levels. A workflow editor has been developed to support the methodology.

1. Introduction

In Software Engineering (SE), the *evolutionary prototyping* is a software development lifecycle (SDLC) model in which a software prototype is progressively created for supporting demonstration to customer and for elicitation of system requirements elaboration [15].The evolutionary prototyping model includes four main phases: (1) definition of the basic requirements, (2) creating the working prototype, (3) verification of the working prototype, and (4) improvement of the requirements elicited.

A propagation is defined by an operation type (i.e., creating, deleting, modifying), a source and a target (e.g., an entity model, an attribute model, a relationship model), and a direction (i.e., forward, backward or bidirectional). For instance, creating a class in one model may propagate the creation of another class in another model. If there is continuous feedback

* Tel.: +52 2222283427.
E-mail addresses: joseguga01@gmail.com, jguerrero@cs.buap.mx.

0167-6423/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.scico.2013.07.003

between the evolving design representations, each design activity can benefit from information derived in the other steps. For example, user interface design benefits from task analysis; problems in the task analysis can in turn be revealed during user interface design, allowing benefit to be derived in both directions [2]. We give a taxonomy of these propagations as model-to-model (M2M) transformations in the Human Computer Interaction (HCI).

The evolutionary prototyping model allows us to create working software prototypes quickly and may be applicable to projects where system requirements are not known in advance, new software needs to be created, and developers are not confident enough in existing software development experience.

Evolutionary design extends evolutionary prototyping in the sense that once a system prototype has been sufficiently refined and validated, it may go further in the SDLC from early to advanced design, if model-driven engineering is used as a method for the User Interface (UI) development life cycle. The main purpose of this paper is to settle down principles on evolutionary design of UIs for workflow information systems.

The remainder of this paper is structured as follows: the next section introduces the concept of evolutionary design, its motivation and its definition. Then, the full implementation of this concept is applied on a software tool and exemplified through a case study. Last, we provide our conclusions and outlook to future avenues.

2. State-of-the-art

The multidisciplinary nature of system design can lead to a lot of different works in the literature which will allow developers to validate the quality of the system development in an efficient manner [27,28].

The work of Chong Lee [4] draws from extreme programming (XP), an agile software development process, and claims centric scenario-based design (SBD), a usability engineering process. Extreme programming, one of the most widely practised agile methodologies, eschews large upfront requirements and design processes in favour of an incremental, evolutionary process. In [1] we can see how different UI design artefacts can be linked to expose their common elements, facilitating the communication on which co-evolutionary design is based. The Clock architecture language [9] was designed to support the evolutionary design of architectures for interactive, multi-user systems. In [35] authors discuss an approach for linking Graphical User Interface (GUI) specifications to more abstract dialogue models and supporting an evolutionary design process, which is supported by patterns. Additionally, a proposal is presented on how to keep connections between concrete user interface, abstract user interface and a task model. Based on the evolutionary character of software systems, Rich and Waters [26] have developed an interactive computer aided-design tool for software engineering; it draws a distinction between algorithms and systems, centring its attention on the support for the system designer.

Model-Based User Interface Development Environments (MB-UIDEs) show promise for improving the productivity of user interface developers and possibly for improving the quality of developed interfaces. There are many MB-UIDEs that follow a formalized method, but their supporting tools do not provide facilities to change the sequence of the method activities, thus restricting the possibilities to adapt the method. The TEALLACH design process [10] aims to support the flexibility for designers lacking on existing environments by providing a variety of routes in the process; from one entry point, the designer/developer can select any model to design independently or associate with other models. Mobi-D (Model-Based Interface Designer) [24,25] a comprehensive environment that supports user-centred design through model-based interface development. Other MB-UIDEs as GENIUS, MacIDA, TRIDENT, Mecano and TADEUS are presented in [10].

3. Designing user interfaces of workflow information systems

In this section, we outline the steps of a method for designing GUIs of workflow information systems (WfIS), called FlowiXML [11,14] that is advocated to automate business processes, following a model-centric approach based on organization's requirements and processes. FlowiXML's methodology [12] applies to (1) integrate human and machine based activities, in particular those involving the interaction with IT applications and tools; (2) to identify how tasks are structured, who performed them, what their relative order is, and how tasks are being tracked. For this purpose, Workflow is recursively decomposed into processes which are in turn decomposed into tasks. Each task gives rise to a task model whose structure, ordering, and connection with the domain model allows the automated generation of corresponding UIs in a transformational approach. A method in evolutionary design is composed of a series of propagations. In Fig. 1, forward and backward arrows denote the propagation of information from one model to another. For instance, a new task model must make available tasks for a process model and vice versa, and a new task in a process model might be detailed with a task model. Jobs, users and organizational modelling just affect the workflow model. Then the workflow model makes them available for process modelling and task modelling. This particular aspect of concepts propagation was significantly useful for the software tools that support the methodology. The methodology consists of the following major steps:

1. WfIS requirements. This is the result of the elicitation of the organization. We assume that there are means such as interviews and direct observation to collect information that will serve as input to identify workflow elements. This step corresponds to the requirements of the system.
2. WfIS design. This step includes modelling of workflow, organizational units, jobs, users, processes, workflow allocation patterns and tasks. Then, mapping this workflow specification to a WfIS.