Chapter 1.13
Software Engineering and HCI

Shawren Singh
University of South Africa, South Africa

Alan Dix
Lancaster University, UK

INTRODUCTION

Technology Affecting CBISs

As computer technology continues to leapfrog forward, CBISs are changing rapidly. These changes are having an enormous impact on the capabilities of organizational systems (Turban, Rainer, & Potter, 2001). The major ICT developments affecting CBISs can be categorized in three groupings: hardware-related, software-related, and hybrid cooperative environments.

Hardware-Related

Hardware consists of everything in the "physical layer" of the CBISs. For example, hardware can include servers, workstations, networks, telecommunication equipment, fiber-optic cables, hand-held computers, scanners, digital capture devices, and other technology-based infrastructure (Shelly, Cashman, & Rosenblatt, 2003). Hardware-related developments relate to the ongoing advances in the hardware aspects of CBISs.

Software-Related

Software refers to the programs that control the hardware and produce the desired information or results (Shelly et al., 2003). Software-related developments in CBIS are related to the ongoing advances in the software aspects of computing technology.

Hybrid Cooperative Environments

Hybrid cooperative environments developments are related to the ongoing advance in the hardware and software aspects of computing technology. These technologies create new opportunities on the Web (e.g., multimedia and virtual reality) while others fulfill specific needs on the Web (e.g., electronic commerce (EC) and integrated home computing).

These ICT developments are important components to be considered in the development of CBIS's. As new types of technology are developed, new standards are set for future development. The advent of hand-held computer devices is one such example.

Copyright © 2009, IGI Global, distributing in print or electronic forms without written permission of IGI Global is prohibited.

BACKGROUND

A Software Engineering View

In an effort to increase the success rate of information systems implementation, the field of software engineering (SE) has developed many techniques. Despite many software success stories, a considerable amount of software is still being delivered late, over budget, and with residual faults (Schach, 2002).

The field of SE is concerned with the development of software systems using sound engineering principles for both technical and non-technical aspects. Over and above the use of specification, and design and implementation techniques, human factors and software management should also be addressed. Well-engineered software provides the service required by its users. Such software should be produced in a cost-effective way and should be appropriately functional, maintainable, reliable, efficient, and provide a relevant user interface (Pressman, 2000a; Shneiderman, 1992; Whitten, Bentley, & Dittman, 2001).

There are two major development methodologies that are used to develop IS applications: the traditional systems development methodology and the object-oriented (OO) development approach.

The traditional systems approaches have the following phases:

- Planning: this involves identifying business value, analysing feasibility, developing a work plan, staffing the project, and controlling and directing the project.
- Analysis: this involves information gathering (requirements gathering), process modeling and data modeling.
- Design: this step is comprised of physical design, architecture design, interface design, database and file design, and program design.
- Implementation: this step requires both construction and installation.

There are various OO methodologies. Although diverse in approach, most OO development methodologies follow a defined system development life cycle. The various phases are intrinsically equivalent for all of the approaches, typically proceeding as follows:

•OO Analysis Phase (determining what the product is going to do) and extracting the objects (requirements gathering), OO design phase, OO programming phase (implemented in appropriate OO programming language), integration phase, maintenance phase and retirement (Schach, 2002).

One phase of the SE life cycle that is common to both the traditional development approach and the OO approach is requirements gathering. Requirements' gathering is the process of eliciting the overall requirements of the product from the customer (user). These requirements encompass information and control need, product function and behavior, overall product performance, design and interface constraints, and other special needs. The requirements-gathering phase has the following process: requirements elicitation; requirements analysis and negotiation; requirements specification; system modeling; requirements validation; and requirements management (Pressman, 2000a).

Despite the concerted efforts to develop a successful process for developing software, Schach (2002) identifies the following pitfalls:

- Traditional engineering techniques cannot be successfully applied to software development, causing the software depression (software crisis). Mullet (1999) summarizes the software crisis by noting that software development is seen as a craft rather than an engineering discipline. The approach

4 more pages are available in the full version of this document, which may be purchased using the "Add to Cart" button on the publisher's webpage:
www.igi-global.com/chapter/software-engineering-hci/29385

Related Content

Adapting a Requirements Engineering Process by Key Factors Estimation
Graciela Dora Susana Hadad, Jorge Horacio Doornand Viviana Alejandra Ledesma (2022). *Research Anthology on Agile Software, Software Development, and Testing (pp. 709-725).*
www.irma-international.org/chapter/adapting-a-requirements-engineering-process-by-key-factors-estimation/294491

Software Engineering Security Based on Business Process Modeling
Joseph Barjis (2010). *International Journal of Secure Software Engineering (pp. 1-17).*
www.irma-international.org/article/software-engineering-security-based-business/43923

Autonomous Drones and Their Integration With AI Technologies
Eugene Berna, Prithi Samuel, S. Ravikumarand K. Vijay (2024). *The Convergence of Self-Sustaining Systems With AI and IoT (pp. 64-84).*
www.irma-international.org/chapter/autonomous-drones-and-their-integration-with-ai-technologies/345506

Teaching Property-Based Testing: Why and How
Isabel Azevedoand Nuno Malheiro (2020). *Software Engineering for Agile Application Development (pp. 230-250).*
www.irma-international.org/chapter/teaching-property-based-testing/250445

Classification of Bug Injected and Fixed Changes Using a Text Discriminator
Akihisa Yamadaand Osamu Mizuno (2015). *International Journal of Software Innovation (pp. 50-62).*
www.irma-international.org/article/classification-of-bug-injected-and-fixed-changes-using-a-text-discriminator/121547