Chapter 7

The Future of Software Development

Karen Church and Geoff te Braake
Port Elizabeth Technikon, South Africa

Software development has changed dramatically in the last fifty years and will continue to change. Its future course is of particular interest to developers, in order to gain the correct skills, and to any person faced with a strategic information technology (IT) decision. It is commonly accepted that computers will play an ever-larger role in modern civilisation. There are many unknowns, but the IT decisions made today will affect the competitiveness and preparedness for tomorrow. Awareness of the central issues that will affect the future of software development is the best form of preparation. This chapter presents a view of the future of software development based on the history of software development and the results of two surveys.

INTRODUCTION

Software development tools and techniques have changed considerably in the last half century, are still changing, and will continue to change in the future as hardware capabilities improve and new technologies make new methods of processing and communication possible.

The aim of this chapter is to draw conclusions about the future of software development from trends that can be identified in its evolution to date. The results of two surveys will help to illustrate some of these trends. The first was a questionnaire survey aimed at software developers which compared their First and Last Project in terms of a number of criteria. The second was a survey of job advertisements in the Computing SA newspaper over a ten year period.

This chapter appears in the book, Successful Software Reengineering by Sal Valenti. Copyright © 2002, IRM Press, an imprint of Idea Group Inc.
This chapter addresses the advancing generations of programming languages which have gained and lost popularity over the survey period. The evolution of coding styles and software architecture will be briefly described. The growing importance of user interfaces will be explained, in addition to a brief description of the increasing complexity of applications from user and developer perspectives. The final section will describe the future trends that can be projected from these points.

LANGUAGE GENERATIONS AND USAGE

The first applications of computers were to gain some form of military advantage based on doing many mathematical calculations very quickly (Arnold, 1991, pp.32-35). Computers then began to be used in business to speed up administrative tasks (Leveson, 1997, p.130). Online transaction processing and later, the personal computer, introduced a whole new dimension to computing by allowing people without programming training to use computers.

The challenge for software developers is to create programs that enhance the lives and work of those who use them. This section begins by describing the software development evolution. The development of programming language generations and their usage is addressed.

LANGUAGE GENERATION

In the early generations of programming languages, machine and assembly languages, the code was written at the level of machine instructions. Many statements were needed to accomplish simple calculations. Programs were long and errors were easily introduced, but difficult to identify and remove.

High level languages (HLLs) were developed to hide the details of implementation from the programmer. This is known as abstraction and is a common theme in the history of programming languages (Watson, 1989, pp.4-10). Each HLL command is translated into any number of machine instructions. HLL coding is shorter, and programs are easier and quicker to write and debug. The commands are fairly easy to learn and meaningful names can be given to variables and subprograms.

Figure 1: Levels of abstraction in Visual Basic and C++

<table>
<thead>
<tr>
<th>Visual Basic</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>frmMain.MousePointer =</td>
<td>HCURSOR lhCursor;</td>
</tr>
<tr>
<td>vbHourglass</td>
<td>lhCursor = AfxGetApp()-&gt;</td>
</tr>
<tr>
<td></td>
<td>LoadStandardCursor</td>
</tr>
<tr>
<td></td>
<td>(IDC_WAIT);</td>
</tr>
<tr>
<td></td>
<td>m_bCursor = TRUE;</td>
</tr>
<tr>
<td></td>
<td>SetCursor(lhCursor);</td>
</tr>
</tbody>
</table>
Related Content

Conformance Analysis of Organizational Models: A New Enterprise Modeling Framework using Algebraic Graph Transformation
[www.igi-global.com/article/conformance-analysis-organizational-models/75464?camid=4v1a](www.igi-global.com/article/conformance-analysis-organizational-models/75464?camid=4v1a)

Reverse Engineering of Object-Oriented Code: An ADM Approach
[www.igi-global.com/chapter/reverse-engineering-of-object-oriented-code/117934?camid=4v1a](www.igi-global.com/chapter/reverse-engineering-of-object-oriented-code/117934?camid=4v1a)

Enterprise Cloud Adoption: A Quantitative Exploratory Research
[www.igi-global.com/chapter/enterprise-cloud-adoption/115444?camid=4v1a](www.igi-global.com/chapter/enterprise-cloud-adoption/115444?camid=4v1a)
Quality-Driven Software Development for Maintenance
www.igi-global.com/chapter/quality-driven-software-development-maintenance/60715?camid=4v1a