Impact of Software Tools and Environment to the Development Process

Christopher Abbas, Sayeed Sajal
Department of Computer Science
Minot State University
500 University Ave W, Minot, ND, 58707
Sayeed.Sajal@minotstateu.edu

Abstract

There's a variety of software tools, with different and distinct functions both to the programmer, and its users; its utilization is pending on the intention of the user. For a programmer, programming tools are specifically to aid and advance their coding processes. A software engineering environment extends a unique suite of productivity to the programmer enhancing his activity efficiency; the environment aids the entire development process in varying settings. The environment is defined by the sort of tools they include and is also distinguished by the relativity between the tools, their integrated functions of the development, and the nature of the task at stake. This paper will highlight more on the specific tools, integrated tool environments including the associated architectures, databases, and parallel and distributed processing issues.

1. Introduction

In the case of computer programming and software product, a development environment is a set of programming processing tools which are been used in the creation of programming software. The term does usually refer to the physical environment as well. An integrated development environment is a development environment that coordinates process and tools to aid the developer with an orderly idea and convenience in the view of writing, testing and reassembling code. An example of most used IDE software is Microsoft Visual studios. The tightly interconnected suite of applications shared between each other on a common database and user interface thereby enhancing programming productivity. A software designing condition stretches out this to programming tools and the entire programming improvement process.

Software tools and situations are intended to improve efficiency. Numerous tools do this legitimately via robotizing or easing some task. Others do it by implication, either by encouraging more dominant programming languages, architectures, or systems or by making the software development task more enjoyable. Still, others attempt to enhance productivity by providing the user with information that might be needed for the task at hand.

2. Integrating Tools in the Environment

Software tools can be merged together as a series of integrated techniques, to link tools that address different aspects of the development process. To state out examples, different tools do support architectural design test case generation. Hence integrating tools has been made to produce a complete environment that bears the entire software development process. Prior environments were either insecure confederation of their tools called single system environment that merge all relevant tools together. Single system environment has its advantages and disadvantages that come with it, one of its advantages is allowing its tools to be tightly coupled to enable the programmer to be alert of the environment and not separate tools. And one of its disadvantages is that they are tightly closed systems which make it difficult to integrate new tools or the use of multiple languages and libraries.

There are approaches that have been used to integrate a single system environment, they include ways for tools to be shared as well as interfaces.

Data integration includes consolidating information living in various sources and providing the user with a bound together perspective on them. This procedure ends up a variety of situations, which incorporate both businesses, (for example, when two comparative organizations need to blend their databases) and logical (joining research results from various bioinformatics storehouses, for instance) domain. “This typically involves the development of a database or repository to hold the information that needs to be shared among the tools. This has the potential to allow a high degree of integration and to simplify the various tools by having them share the work.” [10]. There are few flaws that attach this integration, its that its database does require a huge database structure mandatory to make the environment work. Four most commonly used data integrations are Data Consolidation, Data Propagation, Data Visualization, Data Federation and Data Warehousing.

Another approach to integration is the use of control integration, that comprises of passing messages between tools, information is been shared whenever it needed to be shared or whenever a command is been invoked from another tool. Instead of end-to-end transmitting of information, a central message server is been used as an intermediate means for sharing information. According to [9] “Each of the tools tells the message server what messages it is interested in.” it enables an accessibility to multiple tools as if they are one system, with examples like the use of editor to implore other tools or allow the display of a common focus in repose to the action of the user.

As well as platform integration, the fundamental issue for an integrated solution is that the various tools must be inter-operable. In the past, this requirement implied that all software run on the same machine with the same operating system. Now, though, distributed processing makes it possible to use network-based file systems and network communication to convert and transmit files from one execution environment to another.

3. Software Tools

A lot of software tools have been made from the past years until now. Software tools like visual studios, Notepads and many more have been those that are been used during the programming process. It all deals with the writing of new codes and the edit-debug assistant that aid you in debugging your code to figure out where major and minor errors are situated in your code. They comprise of:

Compilers: It's a special program that converts codes writing in a certain coding language into machine language. During execution the compiler first parses all the languages statement and conditions one after another, all the way through each successive stage, builds the output code, ensuring that every statement and conditions links correcting to end resulting code.

Linkers and Loaders: These unite compiled files with all its libraries into executable files. The Linker is the tool that merges the object file s produced during separate compilation while the Loader is the part of the operating that bring an executable from its residing memory and get it running.

Program Editors: It is basically designed for the purpose of editing of source code or computer programs. Text editors are made with language knowledge that provides them the ability to automatically find or highlight, parenthesis checking and even simple crossing checking. It does ease excess typing buy giving hint one objects and properties you might want to make us off.

Debugging Aids: It is a tool that gives you run-time to be able to find bugs with running programming beyond the time offered by the programming language. We could also define debugging to be the finding and reducing of bugs in a computer program, debugging goes way beyond compile in search to find legal constructs that would be likely potential errors.

Source-Level Debuggers: The user has every control over all execution of the program by setting a breakpoint on any line of the code also makes it possible to step through execution a line at a time and set **source-level** breakpoints. "These have been augmented with visual displays of program values, allowing the programmer to see the application's data structures in their conceptual form." [9].

System Builders: A model of the system that shows a system that should be built or rebuilt. The tools are able to build the original program system and also update a system incrementally based on a set of its source file in respect to changes.

Version Manager: This tool allows more than a single version of a source file to exist at the same time. It grants permission for parallel system development which gives multiple programmers the chance to cooperate without boundaries even with the older released version of the system to be maintained while a newer version is been developed. The version managers do manage different versions of languages and environment. Which lets you install any version and switch between version at any time

Code Generators: These are high-level languages that give a programmer an edge to interactively specify a large portion of a system without having to put it into code. They are used to define user interfaces and interaction of the program with a database system.

Testing Aids: These are tools that help in the process of executing a system with the intent determining if it succeeded or failed and reporting the result. They extend from test case generators that break down source code or determinations to produce a suite of experiments.

Design Editor: It gives the user the ability to design a system using a different type of graphical design**.** "Many of these tools generate at least a code framework based on the design. If enough information is provided with the design, some of these tools can simulate aspects of the system, allowing the developer to test the design at a high level." [9].

Requirements planning process activities	Small project: 5 features						Large Project: 80 features					
	Without tool			With tool			Without tool			With tool		
	Unit effort	Executions	Total effort	Unit effort	executions	Total effort	Unit effort	Executions	Total effort	Unit effort	executions	Total effort
Capture raw requirements	53	12	636	13.5	12	162	248	12	2976	36	12	432
Resolve raw requirements	6.8	23	156	13.2	23	303	6.8	5141	34958	13.2	5141	67861
Resolve features	97.4	5	487	37.7	5	188	97.4	80	7792	37.7	80	3016
Develop release plan	90	2	180	34	2	68	1440	2	2880	544	2	1088
Total effort			1579			757			56411			74688
Productivity	7.6			15.8			3.4			2.6		
Productivity Impact	+107.9%						-23.5%					

Table 1: Requirements planning process activities [4]

4. The Impact of a Tool on Productivity of Projects Using the Advanced Process

*measured in the number of features that can be processed in one 5-day week (2,400 minutes):

$$\text{Productivity} = \frac{\text{Number of features in project}}{\text{The total effort in minutes}} \times 2{,}400 \text{ Minutes}$$

Tools can greatly improve software development processes by facilitating activities not practiced before. For example, a testing tool can help p introduce new testing activities such as branch-coverage analysis. Tools can also help increase productivity by supporting software development activities that are usually carried out with little or no tool support [7]. A tool's impact is not regulated only by its inherent features, but also by the inherited traits of the preceding projects. From the above table, the project executes with tools shows a comparative advantage in the productivity parameter compared to those done without tools.

5. Conclusion

The challenges and realities in applying effective software development tools into project processes are difficult, integrating the tool into the environment to develop a project also poses a major challenge due to the cost of the tools. However, there is a comparative advantage in developing projects with tools, as our study above has shown. Tools advance the development process by increasing your efforts on specific activities or introduces new activities into the process, such as generating and maintaining new data. The effective project development process will succeed by integrating tools to aid the personnel.

References

[1] A. Sulistio, C. S. Yeo and. Buyya, A taxonomy of computer-based simulations and its mapping to parallel and distributed systems simulation tools, Grid Computing and Distributed Systems (GRIDS) Laboratory, Department of Computer Science and Software Engineering, University of Melbourne, Carlton, VIC 3053, Australia.

[2] S. Bendahan, G.Camponovo, and Y.Pigneur, "Multi-issue actor analysis: tools and models for assessing technology environments". University of Lausanne HEC (BFSH1) - CH-1015 Lausanne.

[3] S. E. Lander, S. M. Staley and D. D. Corkill, “Designing Integrated Engineering Environments: Blackboard-Based Integration of Design and Analysis Tools”, Blackboard Technology Group Inc, 401 Main Street, Amhert, MA 01002.

[4] T. Bruckhaus, N. H. Madhavji, I. Janssen, J. Henshaw. The Impact of Tools on Software Productivity. September 1998.

[5] D. W. McIntyret and E. P. Glinert, “Visual Tools for Generating Iconic Programming Environments”, Department of Computer Science Rensselaer Polytechnic Institute Troy, New York 12180.

[6] W. Feng and P. Balaji, “TOOLS AND ENVIRONMENTS FOR MULTICORE AND MANY-CORE ARCHITECTURES”, Virginia Tech, Argonne National Laboratory.

[7] J. A. Edmans, J. Gladman, M. Walker, A. Sunderland, A. Porter and D. Stanton Fraser, “Mixed reality environments in stroke rehabilitation: development as rehabilitation tools”, Division of Rehabilitation and Ageing, Queens Medical Centre, University of Nottingham, Nottingham, UK, School of Psychology, University of Nottingham, Nottingham, UK, School of Psychology, University of Bath, Bath, UK.

[8] J. Grundy , W. Mugridge and J. Hosking, “Constructing Component-based Software Engineering Environments: Issues and Experiences”, Department of Computer Science, University of Waikato, Private Bag 3105, Hamilton, New Zealand, Ph: +64-7-838-4452, Fax: +64-7-838-4155, jgrundy@cs.waikato.ac.nz 2 Department of Computer Science, University of Auckland, Private Bag, Auckland, New Zealand, Ph: +64 9 3737599, Fax: +64 9 3737453, {john,rick}@cs.auckland.ac.nz.

[9] S. P. Reiss, “Software Tools and Environments”, Brown University, Providence, Rhode Island (spr@cs.brown.edu).

[10] A. I. Wasserman, “Tool Integration in Software Engineering Environments”, Interactive Development Environments, Inc. (IDE) 595 Market Street San Francisco CA 94105 USA.