Introduction

Human creations are on an evolutionary path towards perfection, without eventuality. As the activities that astray from this path are abandoned, the new comers wonder how we were able to accomplish so much with awkward and crude tools of the past. Young engineers are fortunate not to have a feel for the agony that we actually went through.

In this article, we will investigate the dimensions of software engineering. All dimensions have been evolving independently. However, the large number of directions taken by attempted solutions has made it difficult to discern a unified evolutionary path.

The scope of our discussion is the practice of software engineering known as application development, in contrast to crafting system software. Below, we briefly analyze the state of each dimension, and then present the Z++ contributions to it.

We then turn our attention to linguistic abstractions of Z++, and the contributions of the Z47 virtual processor. The processor is the infrastructure that absorbs the shock of constant upgrades and changes to the underlying technology.

Z++ is a scientific solution to the most prominent problem of our time that saves money and time as it replaces agonies with ecstasies.

Language, the first dimension

Solving a problem presumes the existence of a language for stating the problem. Disagreements on this dimension have been on the adequacy of pure paradigms such as logic or functional. The wide spread deployment of object-orientation has ended that debate in favor of a more expressive, multi-paradigm language based on types.

C++ is a system level language in the spectrum of middleware. As such, it intrinsically lacks support for distributed applications. An array of activities, attempt to enhance C++ with libraries and protocols towards distributed computing. It is safe to say that all possibilities have been tried, and none have established themselves as the next evolutionary stage for building universal platform-independent distributed applications.

The strength of modern distributed applications rests on the diversity of hardware. The medium for creating such sophisticated applications must evolve from a language with demonstrated expressiveness on every piece of hardware. The language C++ is the only standard that qualifies as a starting point for evolution to distributed computing.

Some languages present a reduced subset of C++ facilities as a compromise for providing mechanisms for distributed computing. It goes without saying that the reduction of established required and familiar mechanisms will result in unforeseen complications when developing large applications.

The language Z++ is a superset of C++ with object-oriented extensions for distributed computing. Z++ executables can move from one piece of hardware to another producing identical effects, while the hardware preserves its unique features. In fact, Z++ can use existing C++ libraries.

The second dimension, interface

An engineer interacts with a library through its Application Programming Interface (API). Thus, in order to determine the suitability of a particular piece of middleware one examines its API. An application, on the other hand, is for ordinary users who intend to use it very much like they use their entertainment center. These users are fairly familiar with Graphical User Interface (GUI) elements.

The same GUI element may appear slightly different from one piece of hardware to another. Nevertheless, all appearances are close enough for users to recognize and distinguish them. Furthermore, the feasibility of a tool allowing an engineer to build platform free GUI is no longer a matter of discussion.

The language C++ dramatically enhances the building of sophisticated API. However, C++ inherently lacks GUI-related statements. Each platform provides its own tools for creating GUI, and requires engineers to remember a large number of counter-intuitive tricks to hook the graphic elements with programming objects.

Z++ presents intuitive expressions for manipulating sophisticated GUI elements. The GUI elements created by the Z++ Visual graphic tool become natural objects within the language, without any specialized tricks or intermediate hooks. The Z47 processor handles the drawing of GUI elements on the destination hardware, taking full advantage of the hardware's features.

Debugging, the third dimension

The statements for validating invariants and raising exceptions are run-time essentials. However, they are not a substitute for debugging. One cannot overstate the value of graphical debugging tools for single-threaded single-platform software.

The tiny code of Z47 virtual processor is identical for all platforms, less the drawing of GUI. Z++ distributed applications can be debugged on a single console. Any combination of interacting diverse platforms can be simulated on a single development platform exactly as the reality itself.

Debugging multi-threaded applications even on a single platform is rather formidable. Thus, the ability of Z47 processor to freeze all threads at a break point provides an immense capability in debugging the least understood behaviors. The freeze action actually yields a deterministic run-time behavior.

It is impossible to exaggerate the significance of a debugging environment that provides an exact snapshot of events and states of a complex heterogeneous network, all on one single console.

The dimension of lifecycle

For the sake of simplicity, we shall only present one aspect of lifecycle concerning the communication between the marketing and the engineering sections of a company. It goes without saying that Z++ carries over all the mechanisms of C++ for managing large projects, and it facilitates the management of distributed applications.

Consider the marketing group using some kind of presentation tool to convey their idea of a new application to the engineering group. This is a much simpler case than suggesting new features for an existing product. In the latter case there will be a lot of disagreement on what can, or cannot be done.

The marketing group can use the Z++ GUI tool exactly like a presentation tool. Let us suppose the marketing has created its wish list for a new application, and stored it in some kind of source control tool.

What an engineer receives from the source control tool is exactly what he/she had to do by continually consulting a presentation tool. Moreover, as development proceeds, any changes to the GUI prepared initially by the marketing group, will in fact get recorded in the source control tool. Since both groups are looking at the same presentation, there is little room for misunderstandings.

The magnitude of the effectiveness of Z++ can easily be seen when the marketing decides to add new features to the product. They will simply retrieve the actual GUI presentations from the source control tool, and discuss their new feature list with the engineers with little ambiguity, if any.

Preliminaries

Before illustrating the value of Z47 processor, we digress to describe our usage of a few terms. This will help us avoid possible ambiguities resulting from the overloaded use of these terms in various contexts.

Loosely speaking, hardware is an operating system (OS) and the physical devices that allow one to interact with the OS. Pragmatically speaking, different versions of an OS may be considered different hardware. Quit often, applications need expensive maintenance in order to continue to run on a new version of an OS.

An application is a piece of software independent of hardware on which it runs. The independence refers to the functionality provided by the application, less the user interface. Thus, the same application, or a variation of it may run on different hardware.

An application is distributed when its execution involves concurrent use of multiple communicating hardware nodes.

A platform consists of a paradigm, a programming language and a collection of tools facilitating the development of applications for a particular hardware. In general, there are a handful of platforms for any hardware.

The Z++ development environment is an abstract platform applicable to any hardware. Nevertheless, following general usage we may sometimes refer to it as a platform independent development environment. In this context, platform is essentially referring to what we defined as hardware. Finally, Z++ Visual is a product based on Z++ abstract development environment.

Z47 Virtual Processor

Perhaps the best way to illustrate the significance of the Z47 virtual processor is to draw a parallelism with a well-known invention. However, the parallelism should not be taken as a form of equivalence of merits.

There are two distinct jump-points in the history of mathematics with regard to numerical computations. The positional representation, which required the notion of zero, naturally lent itself to our present day decimal representation. Then, logarithm boosted astronomical computations.

Limiting our scope to the programs that fall into the category of applications, the notion of object parallels that of zero. The wide spread use of C++, relatively speaking, qualifies the notion of object as a parallel to the decimal representation of numerals. Assuming that one agrees with the parallelism drawn here, the question then becomes, what parallels logarithmic tables.

It is well known that logarithms speed up computations. However, extracting the 20^{th} root of a number is an entirely different story. Here, without the use of logarithms one has no clue as to how to go about doing the computation.

Shifting our focus to application development, presently we are able to port some applications to some hardware, and we can even create some distributed applications. On the other hand, we really cannot use complex type objects in a distributed application. Furthermore, what we can accomplish requires a large number of disparate expertises resulting in expensive and lengthy projects ending in the nightmare of maintenance.

As noted, logarithm cut on cost of doing complex computations, and made it feasible to perform rather formidable ones. A parallel to this would be a platform-free environment that eliminates the prohibitive cost of porting, as well as facilitating the development of distributed software that seem formidable, as things are.

Solutions that are not based on a properly investigated theory are perpetual drain of resources due to their continual radical upgrades. The linguistic abstractions of Z++ are natural extensions of object-oriented notions of C++ constituting a well-researched theoretical foundation. All aspects of the problems being solved have been taken into consideration, in the design of Z++ extensions to C++.

Now, a theory that cannot materialize into an engineering process is not of practical interest. Without the virtual processor, the universality of Z++ abstractions might have been interesting publications. Thus, the Z47 virtual processor is the equivalent of table of logarithms in our parallelism.

Presently, communication via complex type objects in distributed applications is comparable to extracting the 20^{th} root of a number in olden days. In practice, engineers are forced to use extremely simple and flat types. This effectively ruins the actual design of the product in favor of its ability to communicate over a network. The cost of maintenance rises in proportion to unavoidable distortions to the original architecture.

Z++ extends the type system of C++ in several directions. Nevertheless, it imposes no limitations on the type of objects in a function call, regardless of whether the call is internal to the program, or it is destined to a remote node. This uniformity relieves an engineer from having to remember silly details. It also guarantees the permanence of the code developed by the engineer. **Applications written in Z++ are maintenance-free except for developer's own enhancements**.

Conclusion

Many applications of the past were distributed. Indeed, the communication among parts of the application was carried out by physical transportation of tapes or other magnetic mediums. Modern applications simply cannot avoid being distributed. Employees carry a small PDA that interacts with enterprise servers. Sales representatives and agents need the freedom of wireless in order to extend the borders of their office.

The expressiveness of C++, and its availability on all hardware (platforms) has encouraged a large number of partial solutions to the problem of distributed computing. On one hand, some libraries make it possible to port the C++ code to a limited number of platforms. On the other hand, protocols based on XML allow limited communication among a category of applications known as Web Services.

Z++ equips an engineer with expressiveness beyond C++, and can use the existing C++ libraries for web services without the hassles of XML conversion tools. In fact, Z++ opens up the narrow channel of communication to any complex type that an engineer can conceive of.