13/15 rue des Ateliers 1080 Bruxelles, Belgium
Tel.+32-75-235-984, fax.+32-2-414-8404
http://www.imatix.com
info@imatix.com

Technical White Paper

The Pyramid Principle

The main issue in software engineering is the management of complexity. The ***Pyramid Principle*** is a discussion tool that we use when explaining the need for generic solutions to generic problems.

The Pyramid Principle was derived during the 1990's during the author's work with code generation tools, but it applies to many areas and is a valuable educational tool for project managers, technical support teams, and developers.

Copyright

Version Information

Written:	15 February 1998
Revised:	23 January 2000

Disclaimer

The Pyramid Principle

We define the ***Pyramid Principle*** as a method for simplification and abstraction. This model is based on the generally-observed phenomenon that 90% of any system will get 10% of the resources, and 10% will get the remaining 90%. This applies to software development as well as to national economies.

The main application of the Pyramid Principle is to divide a software application into the 10% and 90% parts, then work accordingly. This figure shows the division:

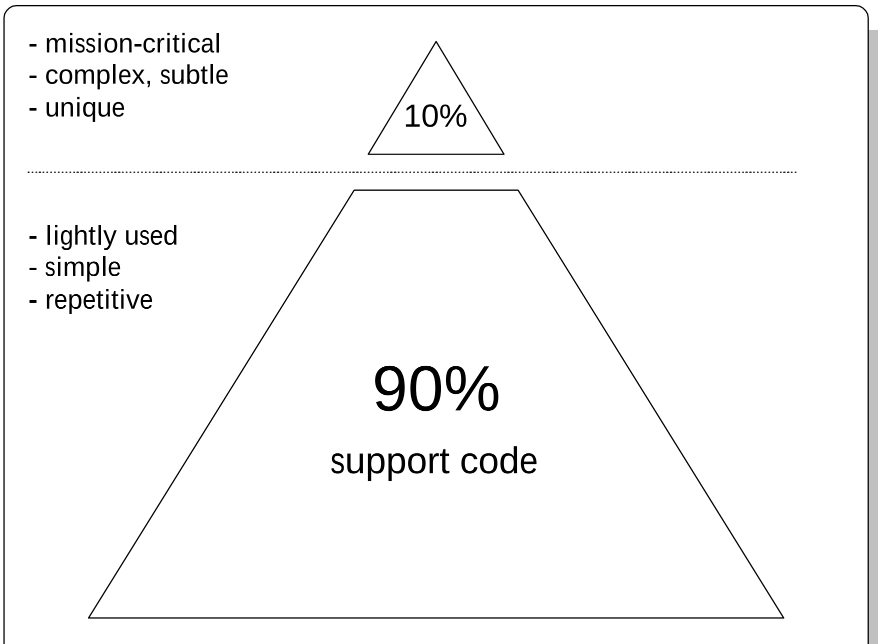

Figure 1: The Pyramid Principle

The ratios may be different (5%-95% or 20%-80%) but in our experience, this division is always possible, and always desirable. By defining this principle from the outset, we can reduce costs and ensure a better result. Many projects pay heavily for treating support code as mission-critical, or vice-versa.

When we apply the Pyramid Principle to application development, we see that:

- In any application, the mission critical programs or 'screens' are a small part of the total source code, but account for a majority of the development effort.
- These programs also account for a majority of system resources (CPU time), bug reports, user time, etc.
- These programs are typically very complex, functionally, and require developers who are both experienced and competent.

It is therefore pointless to try to generate such programs, to simplify them, or to develop them with less skilled developers.

By contrast, when we look at the support code we see that:

- The support programs account for a majority of the total code.
- The support programs are not significant in terms of system usage, largely because the application users invoke these programs very infrequently.
- These programs are typically very simple, functionally, and can be written by less-skilled developers, or generated from templates (naturally such templates ***do*** require the input of a lot of skill).

It is therefore pointless to write such programs by hand (unless this process is guided by strict templates), or to optimise[1] them.

We know that quality and performance are economic decisions. The Pyramid Principle lets us direct the effort at those points where it provides the best value-for-money.

[1] It is well-accepted in software development that 'optimisation' is only positive when applied to 'hot-spots' in an application. In all other places, well-written, but ***not necessarily optimal*** code is most reliable, cost-effective, and appropriate.