Once they have optimised their IT infrastructure, the attention of IT management is now increasingly being drawn towards the price/performance ratio of the applications. Their upkeep and modifications to meet new requirements are a particular drain on money – maintenance and development make up a hefty proportion of the budget. This is exacerbated by the fact that the project cost for applications rises with their age (see graphic). If the budget for maintenance and development has not kept pace, or, even worse, has shrunk the managers in charge will quickly find themselves in a tight corner, with freedom of manoeuvre increasingly restricted.

Here’s the dilemma: the savings potential in this area is significant, but it cannot be realised simply at the touch of a button. As a consequence, concrete numbers are seldom available, which in turn increases the risk of the business objective being missed in the pursuit of savings. In actual fact, assessing the performance of application development and maintenance in comparison with other IT organisations is far from a trivial matter. This is less about measuring the expenses (personnel, hardware, software) which, after all, can be determined relatively easily from the figures available to management. What is rather more difficult is, in the first place, to measure precisely the performance of an application in order to enable comparability of the various applications. A simple comparison of the costs, without any record of the actual performance, would hardly be likely to deliver meaningful results.

The costs of data collection and the benchmark comparison usually turn out lower than commonly assumed. Proven and efficient procedures for identifying the optimisation potential in respect of the maintenance and development of applications are now available:

- **Function Point Analysis**: Standardised application parameters and influencing factors are calculated, with the normalised result of the function points (FPs) forming the basis for the detailed comparison.
- **Backfired Function Points**: Lines of code are converted into function points ("back-
Comparison with peer organisations

Example 1: Cost per function point
- 300,000 lines of code in Cobol
  - Conversion factor as per Capers Jones: 106.7
  - Result: 2,812 function points
- 100,000 lines of code in Java J2EE 1.4
  - Conversion factor as per Capers Jones: 17.8
  - Result: 5,618 function points

Example 2: Number of change requests per 1,000 function points
- Customer: 84.3
- Peer Group: 69.4
  - Difference: +15%

Cost drivers

Once the performance of application development and maintenance have been quantified, general parameters of the application development must also be determined in order to ensure a fair comparison. To give an example, the complexity is influenced by the number of programming languages, the production environment and the processes. Service levels, such as the time to market or the number of errors in the test environment, have an impact on the result as well.

The same applies for development procedures and methods (e.g. structured/procedural vs. object-oriented, waterfall vs. RAD). The maturity of the technical concept and additional requirements after the start of programming also have a considerable bearing on costs and productivity.

Benchmarking projects indicate that application development and maintenance are significantly influenced by these different parameters, which means that cost and productivity comparisons for development and maintenance must always be kept separate.

Example 1: Cost per function point
- 462 €
- 428 €
  - Difference: +8%
Example 2: Number of change requests per 1,000 function points
- 84.3
- 69.4
  - Difference: +15%