

Object Management Group

Framingham Corporate Center
492 Old Connecticut Path
Framingham, MA 01701-4568
U.S.A.

Telephone: +1-508-820 4300

Facsimile: +1-508-820 4303

Benchmark PSIG

White Paper on Benchmarking

Version 1.0

OMG Document bench/99-12-01
December 27, 1999

Editor: Pekka Kähkipuro, *University of Helsinki*

Contributors: Adam Buble, *Charles University Prague*
Anil Gopinath, *Sprint Corporation*
Arvind Kaushal, *Sprint Corporation*
Chanaka Liyanaarachchi, *University of Kansas*
Clark Readler, *MITRE*
David Flater, *NIST*
Douglas Niehaus, *University of Kansas*
Francesco Caruso, *Telcordia*
František Plášil, *Charles University Prague*
Kimmo Raatikainen, *University of Helsinki*
Louis-François Pau, *Eriksson*
Madeleine Chung, *France Télécom*
Pekka Kähkipuro, *University of Helsinki*
Petr Tuma, *Charles University Prague*
Sridhar Nimmagadda, *University of Kansas*
Steve Tockey, *Rockwell Collins*

1 Introduction

Developing CORBA systems that meet performance requirements at application, service or at another level imposes a practical need to be able to compare software parts as to their contributions to the performance of the software system. Moreover, the possible procurement of software, such as an ORB product, to implement these systems imposes similar requirements. The issue is to be able to define generic performance assessments that are tied as little as possible to specific application architectures, platforms, or end user requirements.

While any specific application will of course in most cases have its unique requirements, many developers want to be able to compare specific CORBA technology elements before having built a complete CORBA system, but in such a way that they can evaluate the CORBA technologies in conditions similar to the way they will later be used in the overall CORBA system.

In other cases, users or potential developers of CORBA systems want to evaluate selected CORBA technologies from the point of view of their overall adequateness to an application domain represented by a collection of generic or typical tasks found in the systems deployed in that domain. Such users or potential developers do need measurement procedures allowing them to conduct the comparative evaluation of existing CORBA technologies, without this happening in the context of a given specified application.

In yet other cases, some integrators of CORBA systems which have opted for a specific CORBA technology, e.g. an Object Request Broker implementation, may want continuously to compare that chosen implementation with alternatives in view of possible replacement or upgrades, with an impact on the overall system performances. Such integrators usually cannot undertake the task of integration of alternative CORBA technologies in the existing application, and will want to conduct a comparison using benchmark algorithms representing the specific requirements put on the CORBA technology within the larger system.

Moreover, once a particular set of CORBA technologies has been selected, software engineers may wish to conduct benchmarks in order to obtain additional information for supporting the software engineering process. This information allows them to predict the performance characteristics of the system under design and to avoid design choices that may lead to an unacceptable performance level. This kind of benchmarking is commonplace in projects where performance engineering is part of the overall software development process.

Finally, there is the case of those users or developers who have not opted for CORBA technologies altogether, and who wish to compare them with other technologies achieving possibly in part similar aims at a functional level. Such users and developers will specify benchmark algorithms representing this functional level requirement, in a way which is totally independent from CORBA or OMA concepts, in order to compare the relative performances of CORBA and other technologies.

2 Goals of This Document

The purpose of this document is to provide a distilled view on the responses obtained from the first Benchmark RFI [bench/98-05-02]. In addition, this paper is a first step towards a uniform benchmarking methodology to be applied for various OMG technologies. The goal for the benchmarking methodology is to provide a set of elementary rules to be followed when planning, conducting, and reporting benchmarks. The focus is on the most essential and well-known metrics and techniques. In particular, the Benchmark PSIG is not trying to find solutions to the “eternal benchmarking problems”.

This paper is trying to reflect the views of all parties concerned. In particular, the following views are reflected in this paper:

- For end users, benchmarks provide a tool for selecting the most appropriate set of CORBA technologies for each particular use,
- For software developers, benchmarks are a tool for understanding the possibilities and limitations of the selected CORBA technologies,
- For ORB and service vendors, benchmarks provide a tool for promoting CORBA technology and for choosing the right target clientele,
- For system integrators, benchmarks provide assistance in the selection of appropriate technologies and products for the overall system.

Currently, the primary scope of the Benchmark PSIG is on ORB benchmarks. However, it is expected that CORBA services will be the target of future benchmarking and this has been taken into account in this document.

3 Vocabulary and Definitions

This section defines a basic vocabulary to be used in benchmarking related activities within OMG. The vocabulary has three goals. Firstly, the use of common definitions may prevent unnecessary discussions where the issues are raised only by conceptual misunderstandings. Secondly, the vocabulary provides a quick start for those who are not acquainted with benchmarking. Thirdly, the vocabulary reveals in a concise form relevant benchmark issues as seen by the Benchmark PSIG.

Application requirements: this is a set of user specified requirements for a system satisfying user needs, these including some performance goals and requirements; amongst these requirements some are functional, some are qualitative, and other are quantitative.

Benchmark: a procedure whereby to compare quantitatively different software systems.

Benchmark algorithm: an open, published, software procedure, data manipulation, communications handling or computational algorithm independent of a given implementation platform, independent of any CORBA technology implementation, independent of any software system (see “Software system”), and independent of any other software technology (with the exception of the specification or programming language, which must be an ISO standard). It is accepted that a given benchmarking

algorithm has a number of quantified parameters, which must be defined in a self-contained way using only the standard specification or programming language. Usually, but not necessarily, a benchmark algorithm will need some data to feed it (see “Test data”).

Calibration: when one feature of the platform configuration is changed, the calibration procedure is a procedure by which one selects one performance criterion and measurement (with the feature in its default resp. changed values) by which estimated performance measurement values can be obtained for other performance criteria, by extrapolating to the latter the impact of the changed feature from its measured impact on the selected performance criterion.

Maximum sustainable performance measurement: for a given performance criterion, and a given triggering mechanism (see “Performance criterion” and “Triggering mechanism”), this is the value of the performance measurement at which the system under test (see “system under test”) reaches an asymptotic stable value, or exhibits a significant discontinuity on any graphical plot; in all cases, the reporting of a maximum sustainable performance must always be accompanied by the label (asymptotic stable value, or discontinuity).

Measurement procedure: this is an open published procedure which specifies exactly how, in which order, and with which controls, can be completed all steps whereby, for a given system under test (linked to a given application), a given platform, a given set of performance criteria, these can be measured with specified test data for a given benchmark algorithm (with its parameters) on the system under test.

Parameters: see “Benchmark algorithm”.

Performance criterion: this is a labeled quantitative indicator coming from the application, and which can be defined uniquely on both the software system, and the related systems under test. Possible performance criteria include, but are not restricted to, qualities like response time, round trip time, and throughput (see the respective definitions).

Performance measurement: this is the value of a performance criterion when measured according to a measurement principle, using suitably defined test data and benchmark algorithm, on a specified platform, on both the software system and related systems under test.

Performance measurement procedure: this is the set of software modules, external to the software system, whereby the performance measurements can be computed or collected, and stored.

Platform: this is the combination of processor architecture, processor(s), operating system, run-time library, compilers, linkers, communications interfaces, user interfaces, etc. which are necessary for the compilation, linking, interfacing and execution of the software system and of the systems under test.

Response time: this performance criterion expresses the time between the presentation of an input to the system under test and the appearance of the corresponding output. For an operation invocation in an ORB environment, the response time is meas-

ured from the moment the invocation is made to the moment the invocation returns to the caller.

Round trip time: this performance criterion expresses the response time of an operation invocation (see “Response time”).

Software system: this is a set of software modules not found in the platform, and representing the full implementation of a given application; a given software system may include, or not, software modules which are compliant with CORBA specifications.

System under test (SUT): this is the sub-set of the software system on which the performance measurements are carried out, plus always a benchmarking algorithm, and one or several performance measurement procedures; the benchmarking algorithm may then replace eventually a set of software modules from the software system satisfying similar functional requirements; when the software system incorporates CORBA technologies, and when the performance benchmarking covers ORB performances, the system under test will include all software modules of that ORB; when the execution of the benchmark algorithm requires the interaction of different parts of the software system, these parts must be part of the system under test.

Test data: this is a set of data (static, dynamic, events, data record sizes, data base contents) which serve as input to a benchmark algorithm, and which are essentially tied to the nature of the application. The way these test data are fed into the benchmarking algorithm is specified by the triggering mechanism.

Throughput: this generic performance criterion expresses the amount of work performed by the system under test during the unit of time. When throughput is used as a performance criterion, it needs to be further specified with an indication of the type of work in question (e.g. operation invocations of a certain type) and with a description of the benchmark under which the measurement was made.

Triggering mechanism/procedure: this is the scheduling procedure whereby test data can be accessed by the system under test for execution; such a triggering can for example be event driven, memory scan driven, record driven, etc. The triggering mechanism is implemented outside the performance measurement procedures.

4 Benchmark Principles

According to [Gray91], domain-specific benchmarks must meet four criteria. They must be:

1. **Relevant:** the measurements must reflect typical operations in the selected problem domain,
2. **Portable:** the benchmarks should be easy to implement in different development environments, system architectures, etc.,
3. **Scaleable:** the same benchmark should apply to small and large systems alike, including different flavors of parallel and distributed architectures,
4. **Simple:** the benchmark and its results should be easy to understand.

These criteria are clearly relevant for most benchmarks in the various CORBA technology domains. However, there are additional criteria that stem from the special character of the CORBA architecture and from the goals of OMG. Some of them are discussed below.

4.1 Openness of algorithms and results

The benchmark algorithms should follow the requirements given in the vocabulary above (i.e. openness, independence of any given platform, etc.). A benchmark algorithm is allowed to use private test data even if the algorithm itself is public. This way, users of OMG technologies can exploit already implemented benchmarks using their private data in order to obtain results that better reflect their particular requirements.

If published benchmark results are to be meaningful, they must be accompanied by full documentation on the configuration and tweaks that are necessary to achieve the claimed result. Besides being necessary for the sake of rigor, this would be a useful resource for users who run afoul of "default configuration syndrome". An example of such a procedure is the "Full Disclosure Report" (approximately 100 pages) required by the Transaction Processing Performance Council [TPC99].

4.2 Use of well-understood and easy-to-measure metrics

The benchmark should concentrate on well-understood and easy-to-measure metrics. Typical metrics that have been used with CORBA systems include

- Average invocation time (i.e. round trip time),
- Throughput expressed as the average number of invocations per second,
- Resource consumption as reported by the operating system (e.g. CPU and memory usage).

The variability of performance metrics is crucial in many application domains. In particular, the reliability of real-time systems may be seriously compromised if there is no guarantee of limited variability even when the measured mean value is well within the required range. Consequently, benchmark reports should always include information on variability, such as a graph indicating the approximate distribution.

4.3 The "maximum sustainable performance measurement" principle

The quoted performance measurements should always indicate the maximum sustainable performance for a set of performance measurements grouped into classes (see the definitions). A benchmark report should contain all other measurements as well, but the maximum sustainable performance measurement is to be reported as the result of the benchmark.

4.4 Normalization of benchmark results

A major problem in comparing benchmark results is that different hardware platforms and configurations (e.g., memory, disk drives etc) will produce different results making comparisons difficult. Further, vendors will try many different ways to optimize

performance, including adding cache memory and putting cache buffers on disk arrays. Therefore, some form of normalization is required to make a valid comparison between different systems.

A well-known normalization factor is the "price/performance" ratio (i.e., the lifecycle cost of SUT as configured for the benchmark divided by the throughput). Assuming a five-year lifecycle, cost would include all hardware (purchase price), software including license charges, and hardware/software maintenance. The cost per request then becomes a measure of scalability and a metric to compare against other systems. This approach is mainly applicable in those cases where the performance can be expressed as a single number. However, these cases tend to be rather rare, as the ORB usage scenarios and therefore also the important performance characteristics often vary significantly.

Another way of normalizing benchmark results is to use the "baseline approach". In this approach, benchmark results are presented together with baseline results that have been obtained from a functionally equivalent system where the CORBA technology to be tested (SUT) has been replaced with a trivial surrogate. In many cases, the relevant baseline can be obtained by replacing CORBA based communication with the underlying transport layer (e.g. TCP/IP). The advantage of this approach is that the published benchmark results are more applicable for software engineering and they reveal some of the effects of the operating environment on the results. However, this approach works well only for CPU speed and network throughput. Changes in other factors (operating system, CPU type, compiler, etc.) produce unpredictable effects.

A third way to overcome the normalization problem is to measure ORB functionality in terms of well-defined *prime actions* (e.g. marshalling, dispatching), and on benchmarking these actions. Using a concept analogous to CORBA interceptors, benchmarking of the prime actions can be based on intercepting *prime points* (points separating prime actions) in the request handling path. If this approach is selected, a standard library of prime action measurements could be defined. Alternatively, an API could be specified for obtaining the prime action measurements directly from the ORB implementation (see the section Instrumentation issues below).

4.5 Treatment of scalability

Scalability is frequently thought of in terms of numbers of users and/or objects that can be supported on either a single node or collectively on all nodes in a system. The exact method of measuring this scalability has been subject to some debate.

At least two types of scalability issues need to be taken into account. One is the scalability of CORBA implementations within an end-system and the other is distributed scalability, which refers to the number of end-systems in a network. For end-system scalability, the number of objects that can be supported by the end-system is a performance measure. For distributed scalability, number of end-systems in the environment of the application scenario is a performance parameter. These tests measure the impact the number of objects in a server has on the ability of CORBA implementations to process client requests efficiently. Scalability is an important factor for CORBA applications, which have to serve a large number of CORBA objects on each network node and a number of nodes in a distributed environment. Scalability is also a

factor when a number of independently developed applications that must share the execution platform.

A possible way to treat scalability is to assume that scalability can be measured by the throughput or capacity of the system. This approach is based on the observation that for a fixed system with a given throughput (e.g., a single node) there is an inverse relationship between the response time and the numbers of clients. In other words, the more clients submitting requests the longer the delay. If multiple benchmarks are conducted with a suitable mix of relevant factors, it may be possible to obtain a set of basic scalability results that can be used for estimating the throughput of different system configurations. It should be noted that for most scalability factors, there are limits beyond which the approximation suggested in the third paragraph will fail. Those limits should be investigated in the benchmarking process.

In any case, there is no way to discuss scalability without also considering other performance metrics. Moreover, no benchmark metric for a CORBA based system is relevant unless it is scalable to some extent. Therefore, the scalability aspect should be present in all benchmarks (independent of the selected metrics). This special requirement is due to the fact that CORBA is mostly used in distributed environments. Accordingly, benchmark results (e.g. throughput) should be presented as a function of how extensive the system is. Depending on the benchmarking algorithm, the relevant scalability factor can be, e.g., the number of objects, the number of clients, or the number of nodes in the system. A combination of multiple scalability factors is relevant for most benchmarks.

5 Related ORB Performance Benchmarking Work

The RFI forming the basis for this document has received submissions representing work or competence in the general area of performance benchmarking of middleware, such as ORB's:

- Rockwell
- NIST
- MITRE
- Ericsson
- France Telecom CNET
- University of Kansas (with Sprint)
- University of Helsinki
- Charles University, Prague

In particular, the US Department of Commerce (National Information and Software Technology Institute, NIST) has over 1997-98 developed a framework and has carried out some specific measurements on a number of commercial middleware products loaded in specific ways.

A number of companies, including Nortel and Ericsson, have conducted performance benchmarking of internally developed or/and commercial ORB's, linked or not to operating systems environments, and subject to application specific loads. Ericsson has also defined a set of generic pseudo code applications aiming at performance testing.

Suppliers such as IONA have for their own development conducted specific performance measurements in relation to specific generic tasks in specific environments .

It was recently in May 1999 claimed by the Japanese Distributed object promotion group, that some performance evaluations were conducted in relation to the IIOP interoperability tests. They have used “tests which are not official verification tests, but which were intended to make the test effective and practical”. The DOPG Transaction working group has also worked on defining test sets and test methodology.

The Transaction Processing Performance Council (TPC) (see references in Bibliography) has published a set of performance measuring tests for transactions which could be used for ORB loading.

In June 16, 1999 OMG announced a Formal test and brand program for CORBA, which involves a set of rigorous test suites for interoperability and compatibility with CORBA 2.1. This raises the issue of how such test environments are specified and the relation of such specifications with performance benchmarking evaluation environments.

IETF had earlier issued an RRC 2544 on Benchmarking methodology for networking interconnects, described in <ftp://ftp.isu.edu/in-notes/rfc2544.txt>.

5.1 Comparison with the TPC

This section briefly presents the approach taken by the Transaction Processing Performance Council (TPC) as expressed in their public documents [TPC98, TPC99], and compares it to the possible approaches available for OMG.

Mission. The mission of TPC is to “define transaction processing and database benchmarks, and to disseminate objective, verifiable performance data to the industry”.

The mission of OMG with respect to benchmarking might be less ambitious, since OMG’s primary scope is elsewhere. In particular, OMG might leave the definition of benchmarks to its members and confine itself to specifying guidelines for defining such benchmarks. The dissemination of results might also be left to OMG’s members provided that this is done according to commonly agreed policies.

Processes. The operation of TPC can be roughly divided into four processes. (1) The *development of new benchmarks* and the updating existing ones is initiated by the member companies and terminated after a member ballot. (2) The *publishing of benchmark results* takes place in the form of full disclosure reports (FDRs) that test sponsors submit for review. During a period of 60 days the FDR can be challenged by any of the TPC members and, thereafter, the FDR becomes accepted if the challenges have been resolved. A FDR can be withdrawn either by the test sponsor or by the TPC under certain circumstances. (3) The purpose of *benchmark auditing* is to give credibility to the published results. All TPC benchmarks must be approved by a TPC certified auditor. (4) Finally, the process of *auditor certification* is needed to ensure the high quality of auditing.

As for OMG, all the above processes must be considered and implemented in some form if reliable benchmarking of CORBA compliant products is desired. An important issue is that of auditing the benchmarks: a sufficiently neutral party is needed for validating the results. Initially, TPC allowed benchmarks to be published without auditing and there was no auditor certification. However, in 1994, TPC introduced the current procedures to give TPC results more credibility.

Another important issue is the existence of commonly agreed policies, such as the principles of *fidelity*, *candor*, and *due diligence* adopted by the TPC. They are needed for ensuring the correct use of benchmark results. Otherwise, the benchmarking of CORBA compliant technologies might turn into uncontrolled “benchmarking” that was experienced in the early years of TPC’s existence.

6 Benchmarking Algorithms

6.1 Library of algorithms

It is appropriate to create a library of reference *designs* for benchmarking. These reference designs would include appropriate reference algorithms. However, due to the variations in available programming languages, language bindings to object request brokers, software environments, and the peculiarities of the specific object request broker products themselves, we feel that it is not necessarily appropriate to create a library of source code for reference applications as too many modifications to benchmark application source code would likely be required to make it operate in any given benchmarking situation. However, recent ORB standardization has improved the possibility to create easily portable benchmarking algorithms that would be tied to a specific programming language only.

6.2 Benchmarking framework

A benchmarking framework refers to a technical environment that contains tools and mechanisms for conducting benchmarks independent of the actual benchmark algorithm to be used. This way, a single framework can provide assistance in producing benchmarks of varying goals. Below are listed some items that could be part of the benchmarking framework:

- Definition of a range of *triggering mechanisms* by which test data are fed to the benchmarking algorithms.
- Definition of a range of performance *measurement processes procedures* running during the execution of the benchmarking algorithm, showing exactly how these procedures will impact the performance measurements, this impact being the smallest possible.
- Specification of where in the platform the *test data* and the benchmarking *algorithm parameters* shall reside during the running of the benchmarks; these data and parameters must be allocated addresses in such a way that their fetching/access overheads do not impact the performance measurement procedures.
- Specification of how the measurement procedure can *be adapted to distributed processing environments* and to multiprocessor platforms, in such a way that minimum changes are made to single processor platforms.

- Specification of how the measurement procedure *accommodates access by the benchmarking algorithm and the trigger procedure* to the test data, by either the operating system, or by the memory management, or by a communication bus, or by an Object Request Broker.

The purpose of the framework is to provide a ready-made implementation for a measurement procedure, so that particular benchmark algorithms can be easily fitted into it.

6.3 Automated tool-based approach

A tool-based automated approach provides an alternative way to easily obtain domain-specific or otherwise tailored benchmarks without maintaining a library of algorithms. This approach is based on using a specification language for describing both the algorithm (with a number of variations) and the execution environment for conducting the set of benchmarks. A set of integrated tools is needed for automating the benchmark procedure, for making the relevant measurements, and for producing the necessary reports. Possible variations in the algorithm include items like

- differences in load patterns and different invocation strategies,
- differences in the structural complexity of IDL interfaces,
- differences in data types,
- variations in the number of objects, proxies, connections, etc.

As most ORB implementations contain ORB-specific features, a generic test suit cannot be used for all benchmarks. For such features, a tool-based approach may be the only feasible way to conduct a series of benchmarks.

6.4 A representative benchmarking environment

To obtain accurate performance metrics, it may be necessary to execute CORBA based systems in isolation. However, it is sometimes useful to include other systems in the benchmarking environment in order to create a realistic background load. This reflects the requirement 2 (relevancy) mentioned in the section on Benchmarking principles. The following background loads have been identified as being relevant in this sense:

- Background network traffic.
- Additional objects at server node.
- Additional clients running background applications.
- Background usage of the name service.

Figure 1 illustrates this approach. If this approach is adopted, there is a need for a set of tools for controlling the background load. Some suitable tools already exist, such as NetSpec developed at the University of Kansas. It is a network performance evaluation tool that can be used for generating and measuring network traffic.

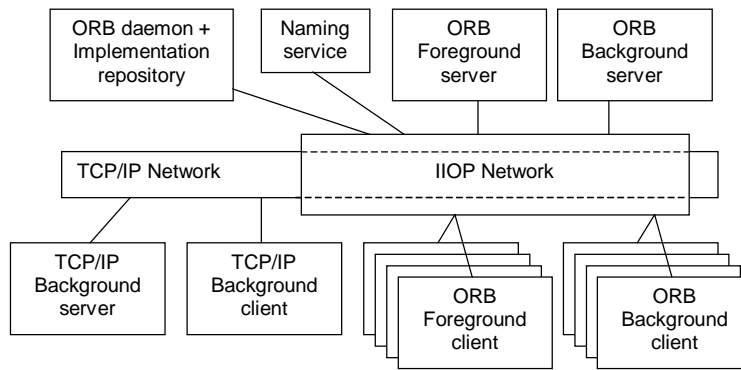


Figure 1. An example of a representative benchmarking environment.

7 ORB Instrumentation Issues

ORB instrumentation refers to specific hooks or APIs that allow software systems to obtain performance measurements from the ORB that are otherwise unavailable for applications. Below are three alternative views for ORB instrumentation.

According to the first view, a *black box* approach is sufficient for obtaining the relevant performance metrics. In other words, ORB instrumentation is not needed because many performance metrics (e.g. response time) can be obtained accurately enough at the application level. Moreover, in some performance sensitive applications it is not even acceptable to have ORB instrumentation due to its negative effect on performance.

According to the second view, a *standard API* needs to be provided to gather information about the performance of core ORB modules. It should be noted that this view is implicitly present in the RFI on Aggregated Computing that poses questions about load monitoring [orbos/99-01-04], and also more explicitly visible in the tentative RFP on ORB instrumentation within the framework of CORBA management [orbos/99-05-07]. There are three well-known arguments against the provision of a detailed low-level performance and instrumentation API. Firstly, ORB vendors might not be willing to open up the implementation details of their products. Secondly, it may be difficult to define a low-level API without unnecessarily restricting the internal implementation of the ORB. Thirdly, “legal” access to technical details within the ORB may prove to be harmful for the portability of CORBA applications that have so far relied on high-level transparencies provided by the ORB.

According to the third view, ORB developers must provide *hooks inside the ORB* at important stages in the CORBA request path so that users can get detailed benchmark data for different criteria. These hooks allow developers to understand ORB limitations, such as the influence of parameter type, parameter length, and the number of parameters on response time. A possible way to implement some of these hooks is to use portable interceptors when they become available; more information can be found from the relevant RFP [orbos/98-09-11].

8 Roadmap

This document will be finalized by fall 1999. There are several possible actions that can be taken either after the finalization process, or in parallel with it. At least the following alternatives are compatible with the views expressed in this paper:

- An RFP/RFI for a benchmarking framework. The result would be a specification of the technical environment for conducting benchmarks, such as the set of allowed triggering mechanism and the recommended measurement procedures. However, the actual benchmark algorithms are not included.
- An RFP/RFI for a benchmarking methodology. This would eventually produce full instructions for planning, conducting, and reporting benchmarks.
- A decision by OMG members to post on an OMG managed Web or FTP site benchmarking algorithms on the legal principle of no-liability, no caution, shareware. This would result in a public repository of generic domain-independent or domain-specific benchmark algorithms to be used within the above framework.
- An RFP/RFI for an automated tool for generating and conducting benchmarks. The tool would use a script and/or a specification language for automating the task of producing benchmarks that comply with the commonly agreed methodology and framework (see above).

All the above items need to be carried out by the OMG in one way or another, as they are an inseparable part of its interoperability, compatibility and branding activities. The only viable long-term solution for the OMG is an active participation in most ORB-related benchmarking activities. The role of OMG should be that of a regulating body, leaving the field open for different benchmark algorithms to be used for different benchmarking needs.

References

- [bench/98-05-02] Object Management Group, Benchmark RFI, OMG Document bench/98-05-02, Framingham, MA, 1998.
- [Gray91] Gray, Jim (ed.), The Benchmark Handbook for Database and Transaction Processing Systems, Morgan Kaufmann Publishers, San Mateo, CA, 1991.
- [orbos/98-09-11] Object Management Group, Portable Interceptors RFP, Request for Proposal, OMG Document orbos/98-09-11, Framingham, MA, 1998.
- [orbos/99-01-04] Object Management Group, Request for Information, Supporting Aggregated Computing in CORBA, OMG Document orbos/99-01-04, Framingham, MA, 1999.
- [orbos/99-05-07] Object Management Group, CORBA Management: ORB Instrumentation, Request for Proposal, OMG Document orbos/99-05-07, Framingham, MA, 1999.

- [TPC98] Transaction Processing Performance Council, Bylaws of the Transaction Processing Performance Council, Version 2.3, San Jose, CA, 1998. Available from "<http://www.tpc.org>".
- [TPC99] Transaction Processing Performance Council, Policies and Guidelines, Version (4.15), 24-June-99, San Jose, CA, 1999. Available from "<http://www.tpc.org>".