



Product Review
SilverStream
Application Server
Version 2.5

A report by
TechMetrix Research

TABLE OF CONTENTS

1. PRODUCT PROFILE.....	3
2. PERFORMANCE MEASUREMENTS.....	7
2.1. INTRODUCTION	7
2.2. RESULTS.....	11
3. FUNCTIONAL EVALUATION	18
3.1. PRESENTATION.....	18
3.2. QUALITY AND RICHNESS OF THE DEVELOPMENT TOOL	19
3.3. HTML INTERFACE GENERATION.....	23
3.4. JAVA GRAPHICAL INTERFACE GENERATION	25
3.5. IDE PRODUCTIVITY FOR SERVER PROCESSING	27
3.6. APPLICATION SERVER	30
4. ANNEXES.....	33
4.1. WORKLOAD METHODOLOGY.....	33
4.2. FUNCTIONAL EVALUATION CRITERIA	37

1. Product profile

Identification	
Product composition	SilverStream Application Server
Release	2.5
DBMSs supported	Oracle, Sybase, Informix, JDBC (type 4) for DB2, JDBC (type 2) for Oracle, and all other ODBC.
Development platforms	Windows 95/98/NT4
Deployment platforms	HP UX 11, Solaris 2.5, Windows NT 4
Editor	SilverStream Software



SilverStream Software, Inc. is a public company (NASDAQ:SSSW) founded in 1996 by a team of specialists coming from such well-known software companies as PowerSoft, Lotus and Watermark. The company's principal product, SilverStream Application Server, offers development and deployment environments for Web applications. Based on Java since its beginnings, SilverStream has been open to the HTML interface in a significant manner since version 2.0. The release discussed in this report, 2.5, is equipped with a polished development environment and offers improved connectivity.

The SilverStream environment covers the entire life cycle of a Web application thanks to some very well integrated tools. SilverStream's developments are based on the relational model; it thus requires only a moderate amount of training and is learned rather quickly. Nonetheless, the development tool lacks some of the advantages associated with object modeling.

Developers used to using PowerBuilder or Delphi will not be thrown off by the development environment. Indeed, thanks to a judiciously designed framework, it is possible to develop a dynamic HTML interface in the same way as an event-driven interface. This hides the technical aspects of the HTTP protocol. This event-driven object paradigm is the originality of SilverStream and it makes for a quick learning process.

However, several points lead us to believe that the tool has not reached full maturity yet. The text editor's functions are much too basic and there is neither a query editor nor a server-side debugger. These shortcomings could hinder development of big projects.

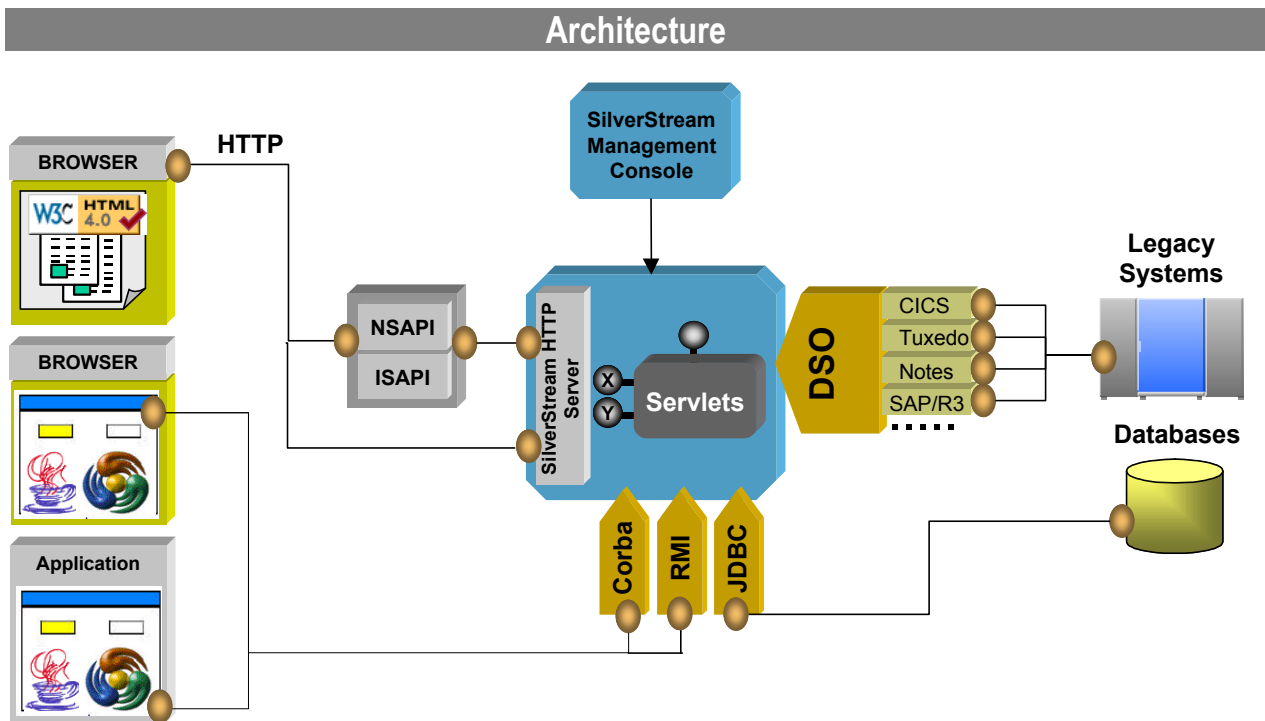
Nonetheless, server-side architecture has everything it needs to become a choice platform for large-scale projects. The extremely rich security model will satisfy even the most demanding administrators. Advanced clustering features make load balancing across several servers possible and offer failover support. A rich technical framework lets the developer trigger actions on calendar, receiving mail, table modification and cluster events.

Enterprise data connectivity has clearly been improved in this version. The main DBMSs, ERPs and IBM middleware on the market are supported, sometimes in the form of separate modules.

With concern for offering operational development and deployment environments, as well as good functional richness, SilverStream has based its environment on proprietary frameworks. In light of J2EE's growing success, SilverStream has announced its support in release 3.0. J2EE brings together Enterprise Java APIs, while preserving the tool's specificity. The portability of SilverStream developments will thus be improved and J2EE skills will continued to be put to good use.

One must be careful when interpreting the good performance and reliability results obtained by SilverStream. The editor developed the application in accordance with our TMBench specifications using Sun's Servlet API and, to a much lesser extent, SilverStream's APIs (for DBMS access). Low-level servlets perform quite well, but require fastidious programming. The application could have been developed in a much more productive and user-friendly manner using SilverStream's IDE. However, in doing so, the editor would have been required to add a software layer that would most likely hinder performance. The editor consciously chose the Servlet API. If we can state that SilverStream is a reliable and effective servlet engine, we cannot guarantee the results of developments when standard use of the product is respected.

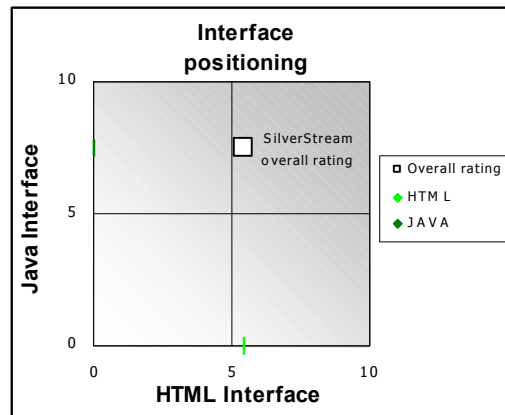
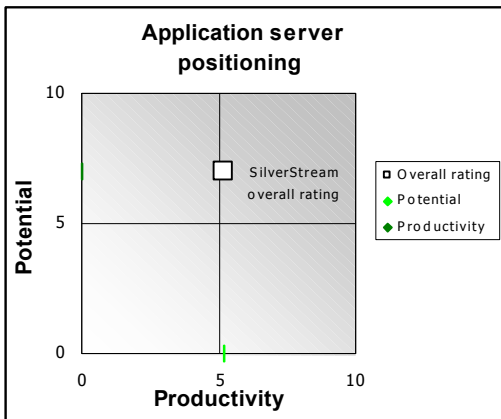
Tool integration and ease-of-learning, as well as strong deployment capabilities, prove SilverStream's determination to provide a product destined to large-scale projects. However, some flaws, which can be attributed to its young age, remain and could hinder large developments.



SilverStream Application Server is open to Java and HTML. It cannot be dissociated from its HTTP engine, which is a spin-off of W3C's Jigsaw and bases its use on servlets. It is nevertheless possible to go through another Web server for the first query, but the following queries will be rerouted toward SilverStream's HTTP server. Server side, the editor offers a DSO interface that makes it possible to access business systems in a unified manner and provides interoperability with the CORBA world via the VisiBroker ORB. Administration is handled by a graphic console or via the management API.

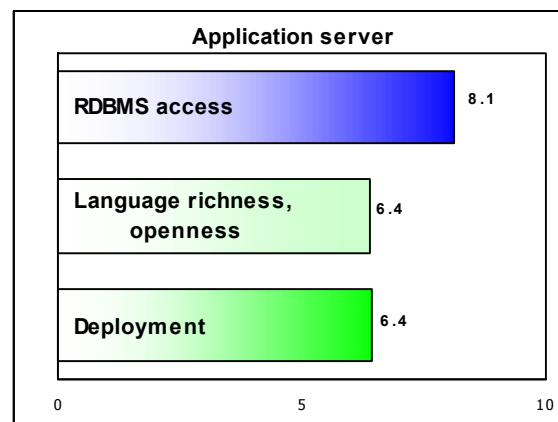
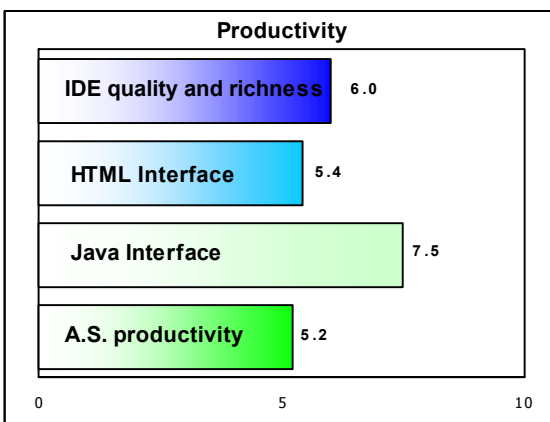
The Pros	The Cons
<ul style="list-style-type: none"> ▶ Effective approach to the development of dynamic HTML interfaces ▶ Choice between Java and HTML on the client station ▶ Carefully laid-out clustering mode ▶ Very rich security model 	<ul style="list-style-type: none"> ▶ No object modeling tool ▶ Mandatory use of SilverStream's HTTP server ▶ Few deployment platforms are supported

Functional Evaluation



By being open to Java and HTML, SilverStream succeeds in entering the interface magic quadrants. Indeed, the environment was originally dedicated to applets and Java applications on the client station, and today finds itself with a polished, visual Java interface editor. SilverStream has only been truly open to the HTML interface since release 2, and although the dynamic HTML page editor is productive and effective, it still has some flaws of youth that result in only an average rating on this axis. Indeed, it is impossible to create headers, footers and templates. HTML support is limited to version 3.2 and the ECMAScript-development assistance is insufficient.

SilverStream bases server-side operation on JDK 1.1 and thus benefits from the richness of Java language. The development tool's excellent integration with the server facilitates the environment's overall productivity. The security model is extremely rich and detailed. Nevertheless, some flaws due to its young age hinder productivity and result in a mediocre rating in this category. These flaws will be even more noticeable during large-scale projects as the code editor is much too basic. In addition, it is impossible to debug server-side applications.



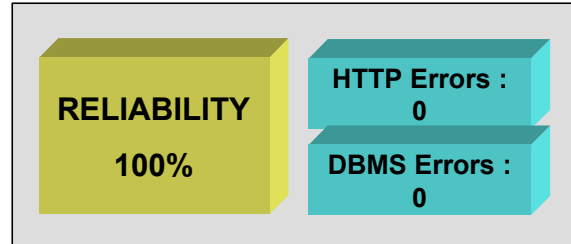
The large number of JDBC drivers shipped with the product and good SQL support make for a high rating in terms of DBMS access. The clustering architecture designed for scalability (load balancing) and reliability (failover) make SilverStream a choice deployment platform. The only things missing are true openness to all HTTP engines and server support for a larger number of platforms in order to widen the tool's capacities. Data source connectivity has been improved (ERP, Notes, CICS...). SilverStream is also open to third-party reporting tools via specialized interfacing. Openness to XML would complete the environment's openness.

Performances

295

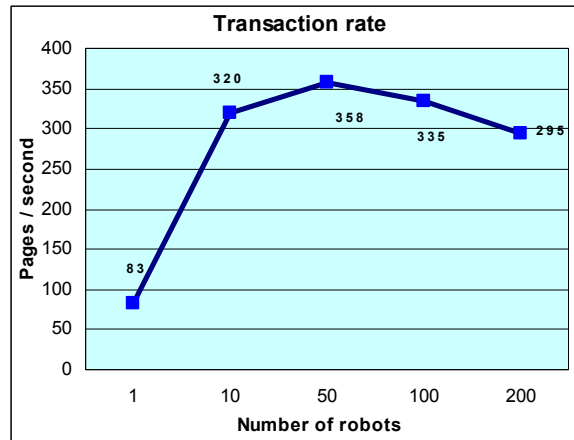
This is the number of dynamic HTML pages delivered per second by SilverStream application server during the running of a “mixed” scenario (stringing together of 8 modules) with concurrent use of the application by 200 robots.

The TMBench-compliant application proved to be entirely reliable running on the SilverStream Application Server, even as the intensive workload placed on the server increased during the tests.



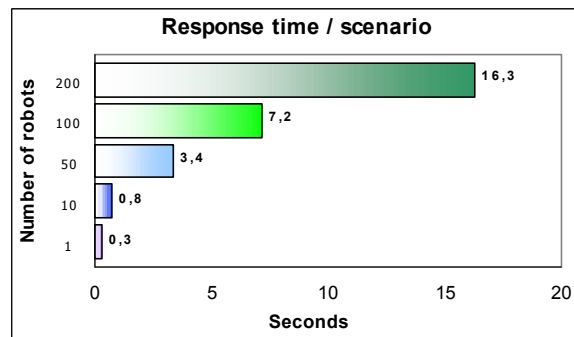
Server configuration was rather particular: three servers functioned in cluster mode on the same machine (a multi-process mode would have been preferable).

SilverStream proves reactive to workload; it presents a high transaction rate.



Server performance decreases when faced with intensive loads (from 358 to 295 pages/second with 50 and 200 robots, respectively), but performance remains more than acceptable.

With SilverStream installed on our platform, 200 robots simulate a load equivalent to that generated by roughly 1000 concurrent users.



The debasement in response time is linear, according to the number of robots. This shows that server resources are used optimally no matter the load.

Technical profile

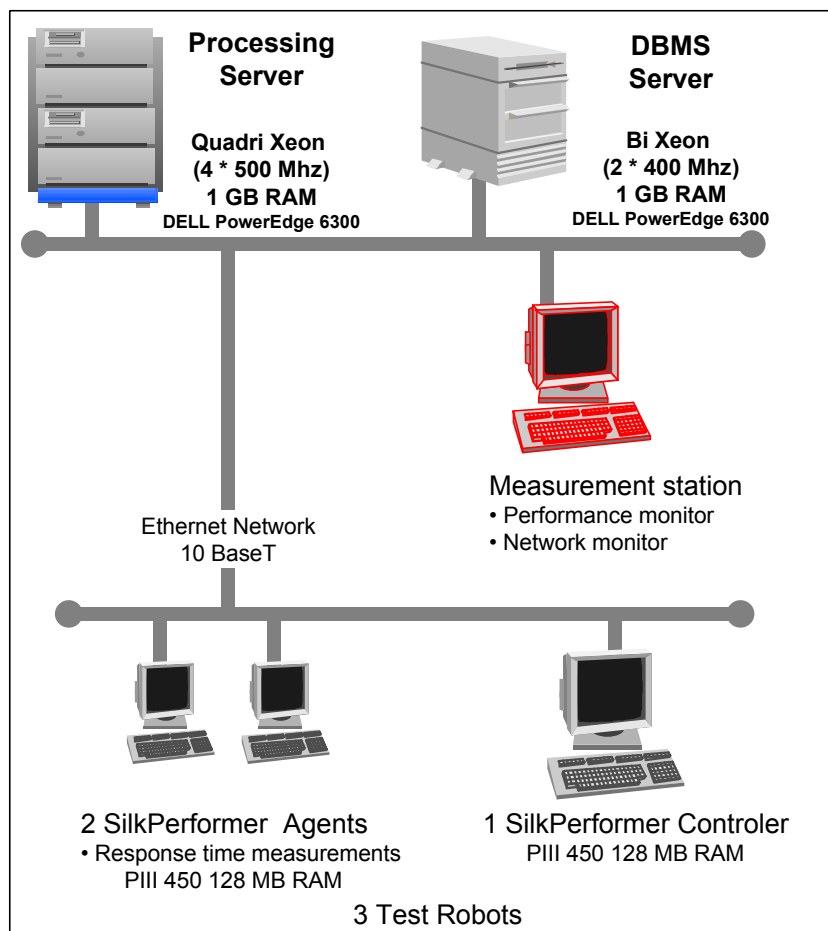
Development license	Approximately \$585 per developer (by 5- and 10- developer packs)
Deployment license	Approximately \$11,580 per processor and per machine
Configuration used for evaluation	SilverStream 2.5 for Windows NT with EDC modules
Complementary modules	EDC (Enterprise Data Connector) SAP R/3: \$5850; EDC Baan PeopleSoft: \$5850 per processor; EDC Lotus Notes: \$3500 per processor; EDC IBM (CICS/MQSeries): \$900 per processor (development license), \$4460 per processor (deployment license); JinfoNet Jreports: \$8500 per processor (deployment license), \$500 per developer

2. Performance measurements

2.1. Introduction

2.1.1. Carrying out of measurements

The performance measurements are carried out on an application developed by the editor. This application is installed on the TechMetrix platform and subjected to sustained workload increases in order to assess and analyze the application server's behavior in extreme conditions. All measurements are carried out by TechMetrix consultants. After that, optimizations made on the premises by the editor are evaluated. The complete specifications of the application tested can be downloaded from the TechMetrix Web site.



Platform used for performance measurements

➤ **Two types of measurements:**

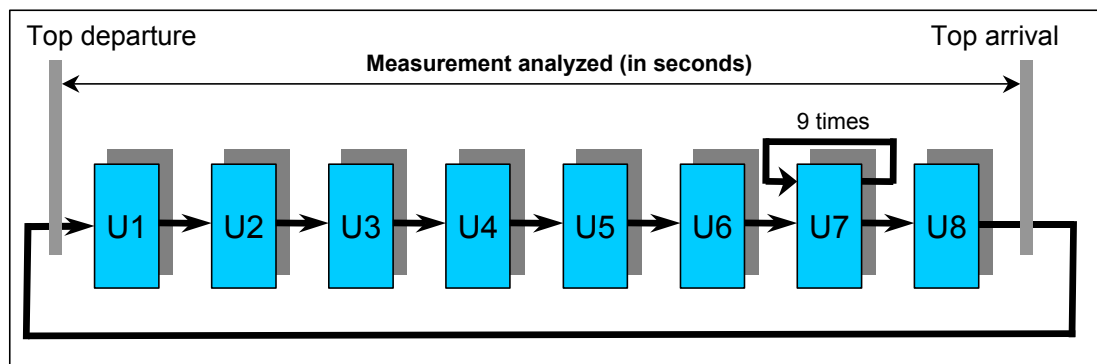
The application is broken up into eight distinct modules (or units), each of which represents one of the main needs of transactional intranet applications. Each model is characterized by its focus on a specific intranet need (simple search, update, calculations...) while remaining as simple as possible in order to put forward, in a precise manner, each of the elements to be analyzed.

A robot, which simulates an application user, carries out the modules in a predefined order several times. The term "scenario" refers to the unitary stringing together of modules; repeated many times in order to obtain an overall average time that is representative of the length of time required to carry out a scenario. During workload increases, each robot carries out the same scenario to simulate a sustained load equivalent to several concurrent users.

The "mixed" scenario:

In this context, the scenario consists in stringing together each module of the application. Each robot carries out the first module, followed by the second module and so on and so forth until the eighth module. Once module 8 is completed, the total time needed to carry out all of the modules is noted. This is the time required to complete this scenario. We should note that step 7, which only consists in storing information in memory, is carried out nine times.

Once the total scenario time has been recorded, the robot carries the scenario out again and again for several minutes. The average time is then recorded. This represents the single-user measurement.



Description of "mixed" scenario with eight modules

For workload increases, the same operation is carried out successively with 10, 50, 100 and 200 robots that concurrently launch the same scenario (sequence of eight modules). The average measurements are then recorded and correspond to the average response time needed per robot to complete a scenario.

The "independent" scenarios:

The "mixed" measurements can have some fringe effects on measurement analysis by saturating part of the application server (memory, HTTP server response, CPU consumption...) and thus resulting in artificially long response times for some modules. To underline the product's real potential regarding certain criteria, workload was only increased in modules 4, 5 and 7. In this context, each robot will repeatedly complete only module 4 several times. Then once the times have been recorded, all of the robots are started again and carry out the next module, 5, and then only module 7.

For each module, load increases are carried out with 10 and 50 concurrent robots. At one instant “t”, all of the robots carry out the same module, in other words, first 4, then 5 and lastly 7.

2.1.2. Analysis of the results obtained

All of the measurements, times, behaviors, etc. are deduced from the physical configuration of the TechMetrix platform and according to the application tested. It is for this reason, that even if this context tries to be as representative of real world business needs as possible, all of the results must nonetheless be situated in our specific context (See the sections Platform and Annexes).

➤ Reliability (HTTP / SQL)

This is by far the most important point, since it indicates whether the application deployed was able to support sustained workload increase without showing sign of weakness. To make this assessment, we analyze the number of HTTP requests that are simulated by the robots carrying out a scenario. We then compare this number of HTTP requests with the number of those actually sent and check the database to see if all the database transactions were indeed recorded.

➤ Measurements (time / transaction rate and throughput)

The measurements obtained are presented in three ways:

- The response time: The average times obtained for a robot to complete a scenario. Detailed measurements for each module are also given for information purposes. These times are those of a robot, in other words, the average time a user will have to wait to complete a scenario in its entirety.
- The transaction rate: Knowing the number of HTML pages that are produced to run a scenario, the number of pages per second is calculated using the average time needed to carry out a scenario. Here, the number presented is the number of pages delivered or the number of scenarios completed by the application server in one second. This measurement gives a good idea of the number of dynamic pages delivered by the application server and of the number of pages that a user receives per second.
- The transaction throughput: Throughput provides information on the processing time of one dynamic HTML page. The “user” axis indicates the latency period for the display of an HTML page while the “server” axis represents the amount of time that the server needs to deliver a dynamic HTML page. As soon as there are several robots, the results on these two axes differ.

➤ **Configuration (database connections, number of processes, number of threads per process)**

Here we provide information on the program architecture that is implemented by the editor to get the most out of his application server. These are interesting points that can be used to judge the application server's potential on this level and to better analyze the results obtained. In addition, they make it possible to assess the potential behavior of a tool when faced with much larger workload increases and identify potential bottlenecks in different physical configurations.

➤ **Consumption (memory, CPU)**

The CPU and memory consumption of the two servers used is presented here. One physical machine is used as both an HTTP server and an application server, while the other acts as the database. CPU and memory consumption make deducing the physical configuration required for the application server tested quite simple, in relation to the number of expected concurrent users.

2.2. Results

2.2.1. Synthesis of the results

The results obtained with the application developed by the SilverStream team should be taken with a grain of salt. The framework used for this development was the Servlet API. SilverStream's API was used – to the slightest extent – for database objects. As it is low level, the Servlet API performs quite well, but denies itself some of the advantages of SilverStream's IDE. Indeed, this IDE offers a WYSIWYG editor for dynamic HTML interfaces, which is based on a higher-level framework. It is also much more productive and user-friendly than servlets and constitutes one of the editor's principal commercial arguments. However, it adds an additional software layer that slows the server down...

In its concern for competition, SilverStream did not rely fully on the development environment (concerning the HTML interface, in particular) in order to obtain better results.

➤ Reliability

Average	1 robot	10 robots	50 robots	100 robots	200 robots
HTTP reliability	100%	100%	100%	100%	100%
Database transactions	100%	100%	100%	100%	100%

During the final tests, the workload tests were finished quickly. However, applying the final touches to the application took a long time and was quite laborious for the following reasons:

- The application was developed with servlets, which required more time.
- Implementing a cluster of three servers on the same machine was tricky. A system for choosing the number of processes would have been better suited to the situation and much easier to put in place.

However, the server had no trouble responding to 200 robots, which represent a load of approximately 1000 real users. SilverStream can thus be considered a reliable server for intensive-use applications.

➤ **Analysis of user-side results**

Average	1 robot	10 robots	50 robots	100 robots	200 robots
Response time per 24-page scenario (in seconds)	0.3	0.8	3.4	7.2	16.3
Response time per page (in seconds)	0.01	0.03	0.14	0.30	0.68

The average response time per dynamic page does not exceed one second, even during extremely intense situations (200 robots). This time is more than acceptable for use in production.

No matter the number of robots, the observed response times are linear. They are very low in a single-user configuration (0.01 second) and increase in proportion to the number of robots. This shows that SilverStream is able to take advantage of the machine's resources, no matter the load.

➤ **Analysis of server-side results**

Average	1 robot	10 robots	50 robots	100 robots	200 robots
Transaction rate (pages delivered per second)	83	320	358	335	295
Transaction throughput (time in seconds between delivery of two pages)	0.01	0.003	0.003	0.003	0.003

Apart from the single-user configuration, SilverStream generates a dynamic HTML page every 3 thousandths of a second in situations with a number of robots ranging from 10 to 200.

Nonetheless, an overview of the number of pages (transaction rate) underlines a debasement starting with 50 robots that cannot be ignored (from 358 pages/second with 50 robots to 295 pages/second with 200 robots).

2.2.2. General characteristics

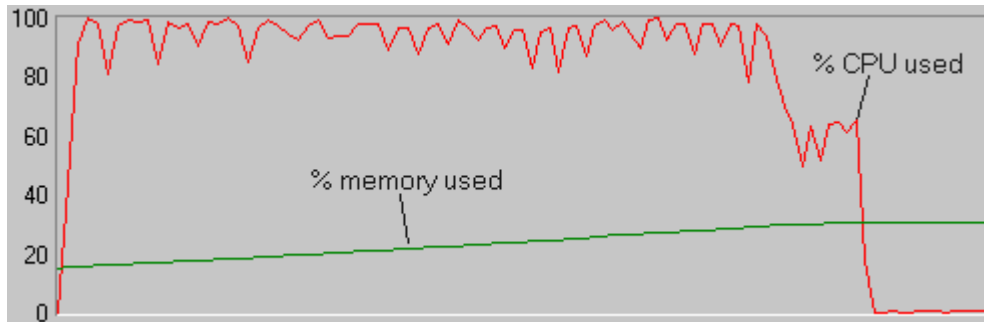
Configuration	1 robot	10 robots	50 robots	100 robots	200 robots
Number of open connections to Oracle database	210 to 218	210 to 218	210 to 218	210 to 218	210 to 218
Number of started clusters (equivalent to the no. of processes)	3	3	3	3	3
Number of threads per instance (or process)	120	120	120	120	120

The performance tests were carried out in order of decreasing load. At the beginning of the tests, the maximum number of connections in the pool was set at 70 per cluster, which makes for a total of 210 (70*3) physical server connections to the database. Oddly enough and despite a maximum of 200 robots, SilverStream needed to open eight database connections in addition to the 210 connections present since the beginning of the tests.

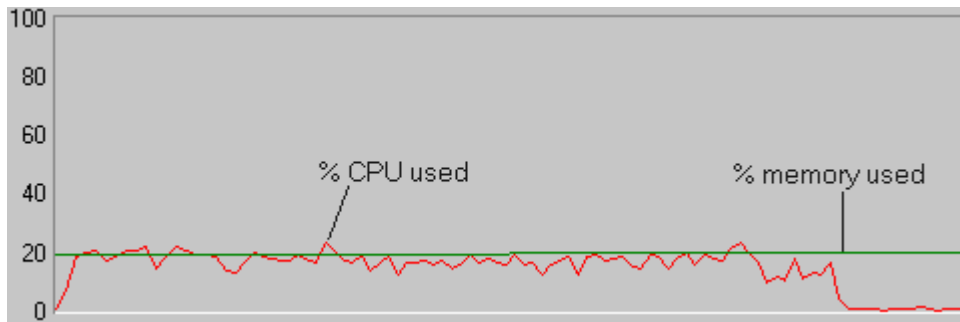
We should note that this configuration, with one connection per user, could create problems in terms of database server memory or the limitation of the number of concurrent, open database-side connections.

Before beginning the tests, the editor chose to set the maximum number of threads per process (cluster, in this case) at 120 and create a safety net for the server. The maximum number of threads really used during the tests was roughly 70 per instance with 200 concurrent robots.

➤ **CPU consumption and memory used with 200 robots**



*CPU consumption (4*500 MHz) and memory used (1 GB of RAM) on the processing server with 200 robots (in %).*

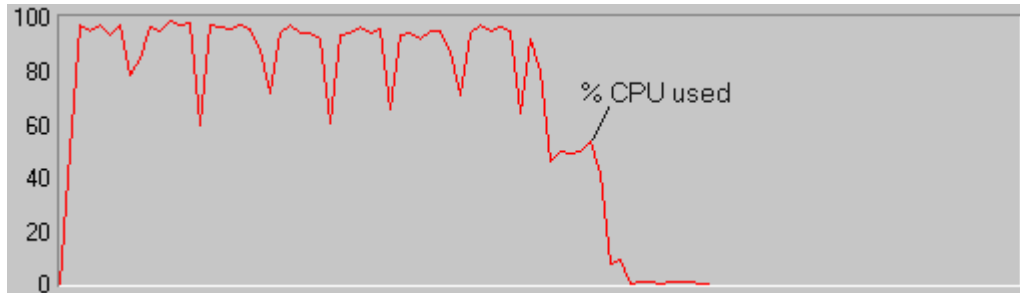


*CPU consumption (2*400 Mhz) and memory used (1 GB of RAM) on the DBMS with 200 robots (in %).*

The above graphs pinpoint the architecture's bottleneck: the application server. Its calculation resources are fully used with 200 robots. However, the memory isn't used very much, only 300 MB of the GB available on the processing server are reserved.

The resources of the database server were used only slightly (only 20% of the CPU and memory). This confirms SilverStream's performance in dialoguing with databases. Nonetheless, an identical configuration using a database such as Oracle 7.x with 218 open connections would have used much more memory.

➤ CPU consumption with 10, then 1, robot on the processing server



*CPU consumption (4*500 MHz) on the processing server with 10 robots (in %)*



*CPU consumption (4*500 MHz) on the processing server with 1 robot (in %)*

The calculation resources are in great demand, even with the most moderate loads. Indeed, one reaches high peaks of 100% CPU use with 10 robots, which is understandable considering that the transaction rate is almost at its maximum with this load (320). The graph on the right shows very little processor activity with one robot.

Consumption of machine resources is consistent with the transaction rates observed with different loads. This tells us that in order to retain the same response times it is necessary to put processes in parallel on several physical servers from 10 robots on.

2.2.3. Table of results

➤ **Average time, in seconds, for each module of a “mixed” scenario**

The eight modules of the scenario allow us to center the analyses on basic features. The role of each module is described below:

- Module 1: large “select” (more than 5000 records returned without cache in a database)
- Module 2: sum of the small “select” without cache in a database
- Module 3: sum of the small “select” with cache in a database
- Module 4: multi-criterion search with dynamic request, without cache
- Module 5: database updates (simple insertion and updates)
- Module 6: algorithm
- Module 7: storage of user context in memory
- Module 8: mass insertion

Number of robots	Modules							
	1	2	3	4	5	6	7	8
1 robot	0.01	0.02	0.00	0.02	0.01	0.12	0.03	0.02
10 robots	0.05	0.06	0.01	0.06	0.07	0.23	0.11	0.12
50 robots	0.29	0.27	0.14	0.39	0.40	0.44	1.01	0.40
100 robots	0.57	0.48	0.33	0.89	0.87	0.69	2.72	0.57
200 robots	1.06	0.85	0.63	2.15	2.05	1.19	7.30	1.01

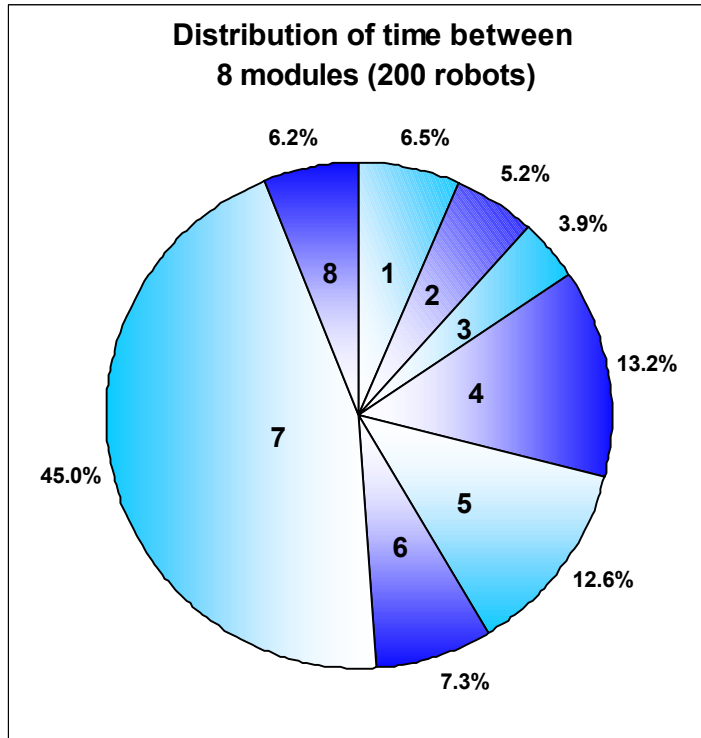
The data in the above table shows SilverStream’s coherent behavior in relation to workload. This behavior results in response times that increase logically with the number of robots that solicit the server. One should note a considerable increase in module 7’s idle time with 200 robots (7.30 seconds) in relation to that with 100 robots (2.72 seconds). This difference underlines some of SilverStream’s difficulties in effectively handling operations that solicit context management strongly.

➤ **Average time, in seconds, for modules 4, 5 and 7 (independent scenarios)**

Number of robots	Modules		
	4	5	7
1 robot	0.02	0.01	0.03
10 robots	0.06	0.07	0.11
50 robots	0.39	0.40	1.01

The response times above confirm SilverStream’s behavior in relation to workload. Idle time logically increases with workload.

➤ **Distribution of time per module**



Distribution of response time between the 8 modules of a scenario with 200 robots (in %)

Step 7 takes up just a little less than half of a scenario (45%). This large proportion is normal as Step 7 is made up of nine successive insertions of data in the context.

For the rest, distribution is relatively consistent, apart from steps 4 and 5, which represent 13.2% and 12.6% of total scenario time. These steps are multi-criterion search and insertion/update of data from a form, respectively. The additional checks carried out during these modules justify the results.

3. Functional evaluation

This chapter is dedicated to the evaluation of SilverStream 2.5 in different functional domains. The quality and richness of the development environment will be looked at first, followed by a discussion of openness to other tools and the application server's potential.

3.1. Presentation

➤ Product description

SilverStream application server is made up of the following modules:

- SilverStream Designer, the development environment
- SilverStream Application Server, the application server
- Microsoft's JVM
- Fulcrum Search Engine, a search engine destined for online help and SilverStream applications. It supports full-text search and indexing in documents and data, and supports more than 50 different types of documents (Office, Lotus, PDF, SGML, HTML...)
- Sybase SQL Anywhere, a database engine

SilverStream requires the use of a database, which has three roles:

- Storing all developed applications
- Storing all sample applications
- Stocking the repository, SilverMaster, which SilverStream uses for internal management.

SQL Anywhere is proposed as database, but any other database may be used.

3.2. Quality and richness of the development tool

3.2.1. Installation

➤ Prerequisites

The development tool can be installed on Windows 95/98 or Windows NT 4.0. The minimum configuration recommended for development is a PC equipped with a Pentium 133 MHz processor, 60 MB of disk space and 96 MB of RAM. In practice, if the developer wants to have sample bases at hand and wishes to comfortably test developments, he should be equipped with a Pentium II with 128 MB of RAM and 300 MB of disk space.

The presence of a 4th-generation browser is necessary if one wants to take advantage of online help and test SilverStream applications that are based on HTML pages and Java forms. It is not necessary to have an HTTP server since one is included in the product.

SilverStream proposes the use of Microsoft's virtual machine, but the use of another (1.1 or 1.2) such as that of Sun or HP-UX is possible if it is housed on the station.

➤ Procedure

The installation of SilverStream went off in three steps, restarting the machine between each step:

- Installation of Fulcrum Search Engine posed no problem.
- Installation of Sybase SQL Anywhere required some tricky configuration that could have been better documented.
- Finally, installation of SilverStream, with the choice between simple and personalized installation, went off well.

Several sample applications can be installed with the server. It is necessary to create a database should one wish to use the test application to become familiar with the product.

➤ Documentation and tutorials

The printed documentation is quite good and often includes code examples. It is made up of the following volumes:

- Installation manual: This volume documents installation and configuration of the server, databases, and EDCs (Enterprise Data Connectors).
- Tutorial: Using pedagogic examples, this volume illustrates the server's functions. The examples become progressively more difficult as one advances in the tutorial.
- Administration manual: This guide documents HTTP-server administration, security configuration and the implementation of the cluster mode for failover and load balancing.
- Programming manual: This document explains in detail how APIs work and how to use them.

3.2.2. The development environment

The development environment, SilverStream Designer, is the central toolset that manages all of the different elements of the application in a tree structure (objects, pages, tables...). It allows developers to create new applications, manage their security, publish them on the application server and test them. SilverStream Designer assigns distinct roles to several development tools:

- The Page Designer is used to build static and dynamic HTML pages in WYSIWYG mode.
- The Form Designer handles development of Java interfaces and business code. It has many characteristics that facilitate data form creation (graphic control lists, link to table fields...).
- The View Designer allows development of Java interfaces that are capable of displaying the data in a base.
- The Table Designer ensures table and column creation. It is also possible to define attributes outside of the relational model: default values, validation rules, textual search or record version management.
- The Relationship Designer displays existing table joins and allows for the creation of new relationships that are managed by SilverStream this time.
- The Object Designer offers a Java programming environment and a CORBA IDL.

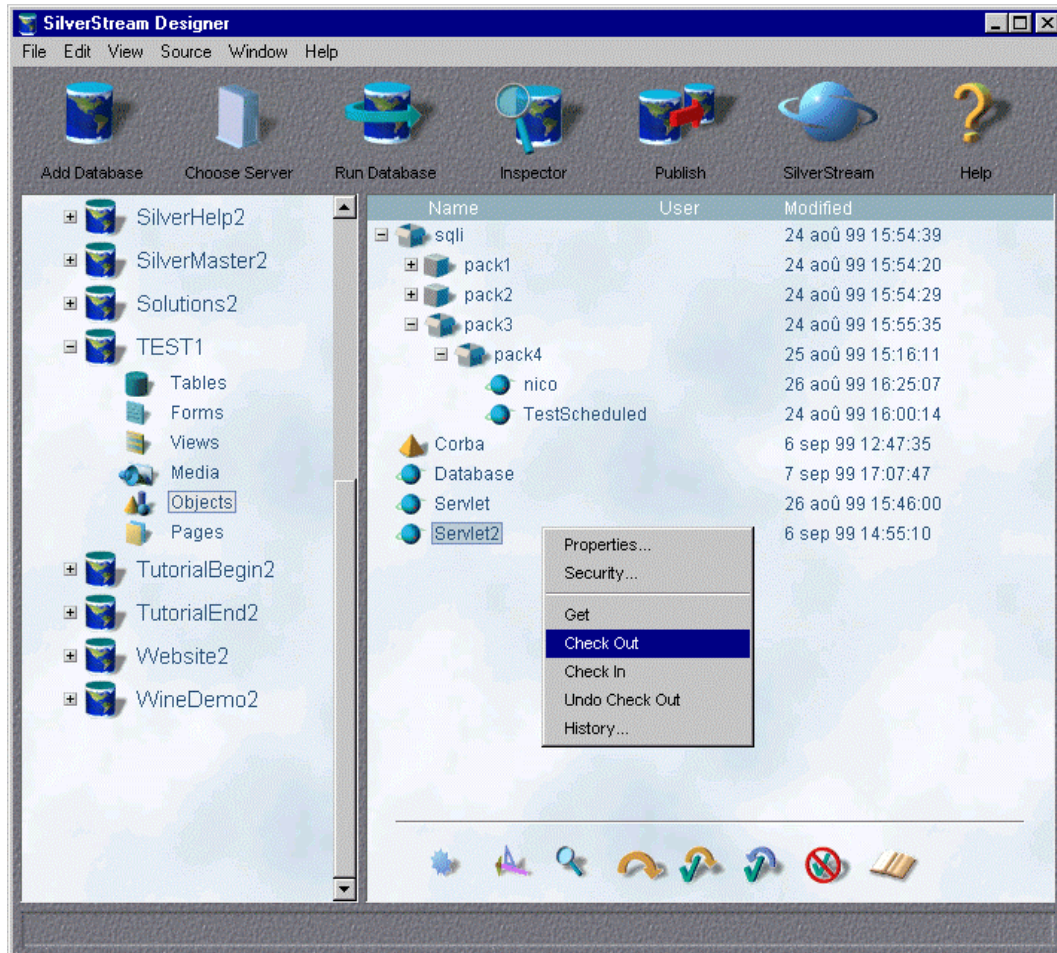
The strong visual identity of these tools, as well as their consistent ergonomics, makes SilverStream Designer a well-integrated development environment that is pleasant and easy-to-use. Numerous wizards allow quick creation of diverse development elements. Even if these elements are not all that helpful when developing complex enterprise applications, they are nevertheless interesting from a pedagogic point of view and contribute to the environment's ease-of-learning.

One of SilverStream's biggest assets is the flexibility and simplicity of developing dynamic HTML interfaces. The development tool uses an object/event-driven paradigm, which makes it possible to build a dynamic HTML interface in the same way as a traditional event-driven interface. This approach, which differs from that taken with traditional scripting tools, proves particularly productive and presents strong similarities with such client-server development tools as Delphi, PowerBuilder and VisualBasic.

Although HTML-page design poses no particular problem, page linking is not displayed graphically. Thus, no global view of the application is provided.

Application modeling goes through table creation and definition of primary/foreign keys. Good integration between graphical interface components and table fields in SilverStream make it a tool adapted to the relational model. There are, however, two weak points: a SQL-query generator would not have been superfluous and the lack of an object-modeling tool hinders the tool in terms of large-scale projects.

SilverStream Designer does not have a native tool for managing development versions, but it does offer a good quality interface with other specialized tools such as Merant Intersolv PVCS or Microsoft Visual SourceSafe.



SilverStream Designer interfaced with PVCS

The recent integration of Jinfonet Jreports makes it possible to create reports and diffuse them in the form of Java applets or HTML pages using SilverStream as the intermediary. This product is sold separately and Jinfonet offers a special SilverStream version, which can be integrated into the server as a servlet. Report development and testing must be done in with the Jinfonet development tool, Jreports Designer. Jreports for SilverStream 2.5 is currently available in beta version.

ReportHeader	
PageHeader	<i>Jinfonyet Gourmet Java</i>
	MM/dd/yy hh:mm:ss
GroupHeader (O	ORDERS ORDER ID XXXXXXXXXXXXXXXXXXXXX
	PRODUCT NAME CATEGORY CUSTOMER NAME UNIT PRICE QUANTITY DISCOUNT ORDER DATE SHIPPING COST SHIP VIA
Detail	XXXXXXXXXXXX XXXXXXXXXXXX XXXXXXXXXXXX ###.### ##### ###.### MM-dd hh:mm ###.### XXXXXXXXXXXX
GroupFooter (Or	
PageFooter	#####
ReportFooter	

Jinfonyet Jreports

SilverStream also ships with its own HTTP engine. The Fulcrum indexing engine allows textual searches in applications. The DBMS, SQL Anywhere, stocks SilverStream metadata and enterprise data.

3.2.3. Evaluation

➤ **Summary**

The development environment is characterized by a good level of integration and good functional richness. Learning to use the tool proves to be quite quick, especially for developers who are familiar with L4G-type event-driven programming.

However, the product remains incomplete on some points. The lack of a SQL query editor is disadvantageous inasmuch as the tool offers relational application modeling. Even though the tool is based on Java, no framework allows true object modeling, which can hinder large-scale projects. In addition, no assistance is offered to facilitate implementation of multilingual applications.

IDE quality and richness	Rating/10
Installation, learning process, simplicity	7.5
Product completeness, openness to external tools	4.5

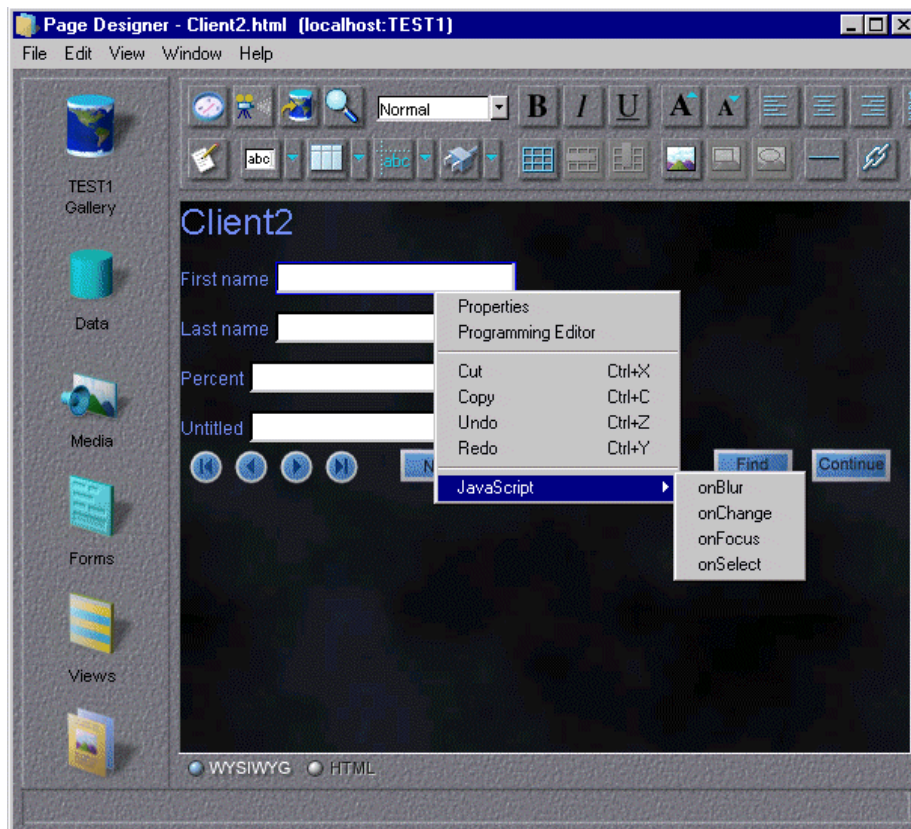
3.3. HTML interface generation

3.3.1. Design and optimization of the HTML interface

The HTML page editor offers a WYSIWYG mode. Page creation is greatly facilitated by the presence of a palette containing standard form fields (static text, text box, button, check box...)

By generating HTML 3.2, SilverStream guarantees development compatibility with all browsers; but denies itself the potential of HTML 4. HTML development is done in WYSIWYG mode or in programming mode. A syntax editor colors tags and allows for code modification. The tool is bi-directional; it is thus possible to integrate HTML code generated by another tool. This code can then be exported in order to be modified. Table and cell manipulation is well supported.

Nonetheless, the HTML editor could be improved. Indeed, it is impossible to create templates; and headers, footers and style sheets are not supported. This can hinder large-scale project development. The workshop allows for the addition of ECMAScript functions (VBScript, JScript and JavaScript), but there is not a client-side library to validate data coherence. Validation rules can be created, but these will be activated on the server.



SilverStream Page Designer

3.3.2. Evaluation

➤ **Summary**

HTML interface generation is handled in WYSIWYG mode and proves to be flexible. But the IDE will reach full maturity only when templates, HTML 4 and automatic server-side script generation are supported.

HTML interface generation	Rating/10
Automation of HTML interface generation	5
HTML interface creation: assistance and potential	5.8

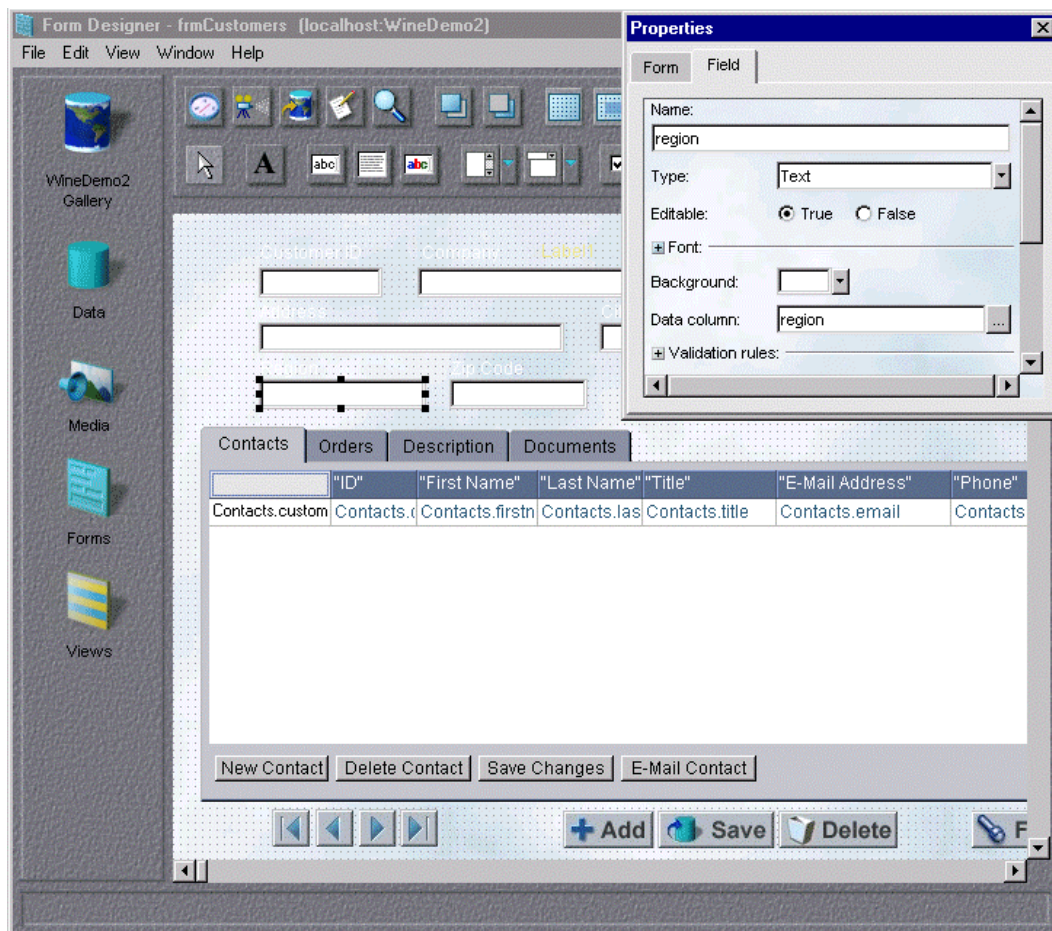
3.4. Java graphical interface generation

3.4.1. Design of the Java graphical interface

With regard to the graphical interface, SilverStream inherits from its first version, which was based on client-side Java architecture. Java interface construction takes place in WYSIWYG mode with the help of a palette proposing standard AWT components, enhanced by proprietary components that make up for AWT's shortcomings. Their behavior is configurable using an object inspector. Numerous wizards generate ready-to-use transactional applications with modifiable code.

Two tools ensure Java interface creation: Form Designer and View Designer, which remain oriented toward creation of forms and data views.

The only thing we reproach is the impossibility of creating JavaBeans. It is necessary to use a third-party tool and later integrate JavaBeans into the SilverStream palette.



SilverStream Form Designer

3.4.2. Evaluation

➤ **Summary**

SilverStream is a mature tool for creating Java interfaces. Nevertheless, one must be careful when implementing this type of architecture. Even if SilverStream encapsulated a flood of RMI's in the HTTP protocol in order to avoid partially deployment problems in version 2.5; there are numerous deployment constraints.

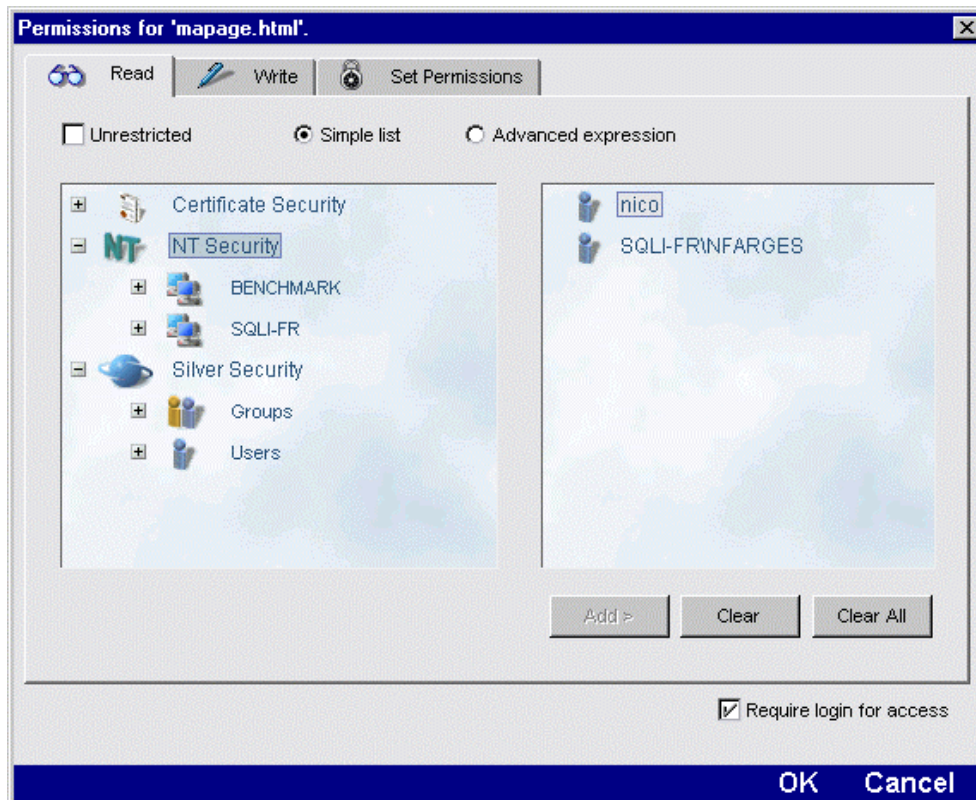
Java interface generation	Rating/10
Automation and assistance in elaborating Java graphical interfaces	7.5

3.5. IDE productivity for server processing

3.5.1. Assistance with server-side code generation

Development of SilverStream applications is based entirely on Java; it thus benefits from all of the advantages associated with Java such as portability, security and object-based development.

SilverStream provides a surprisingly rich security framework. All application elements have configurable security ranging from simple access control to authentication and the most complex rights management mechanisms. For authentication, SilverStream gives the choice of creating a user repository, but also of assigning rights to users defined in other directories (LDAP, NT, NIS+). Authentication mechanisms are based on a simple password or x509 certificate. It is important to note that the security is configurable during development and production.



Security configuration of an HTML page

By integrating the VisiBroker ORB and an IDL code editor, SilverStream is interoperable with CORBA objects. Openness to EJBs has been announced for version 3 of the product. Interoperability with the DCOM world can be ensured if the application server is based on Microsoft's virtual machine. However, lack of assistance for this model risks to render this type of development difficult.

SilverStream offers an object model called Business Objects. Despite its name, this model is not an object modeling framework, but more of a technical framework. Indeed, it allows for the development of technical objects that are capable of intercepting events stemming from the server. It is thus possible to trigger events on mail send and receive, calendar, table modification or cluster events. Once these objects are created, it is possible to bind them to graphical components of the interface using examples in the Page Designer or Form Designer.

The application server handles context management. It is possible to store and remove objects relative to an object session "httpsession" using "put" and "get" methods. So that the right session corresponds to the right client, SilverStream generates a cookie and sends it to the client station at the first connection. This cookie contains a unique session identifier that makes server-side context correspond. Cookies are the only method offered for context management. Automatic transformation in URL long would have made it possible to include the browsers that reject cookies. One must be careful of this characteristic during application deployment.

3.5.2. Tools and automation provided by the IDE to improve productivity

The tool includes a source code editor that could stand some improvement. It helps with completion and offers syntax coloring, but does not allow search and replace in several files. A Java compiler is provided and error messages are given below the source code editor.

Lack of a server-side debugger is a big disadvantage for quick application development. SilverStream has announced that a server-side debugger will be included in version 3.

The server can administer remotely with the help of a graphic console. However, an HTML version of this console would have been a good idea. The console can administer security, specify log files for server events, and tune connections to databases and virtual machine parameters.

The administration console provides server-use statistics such as the name and number of connected users or information on active Java threads. It is also possible to obtain such information about SilverStream's HTTP server as the number of HTTP requests received or average response time. However, graphics would have helped give a better idea of server activity.

3.5.3. Evaluation

➤ **Summary**

The good functional richness of server-side frameworks makes the IDE productive. The security framework is irreproachable during development and production. Numerous ready-to-use technical objects are provided and the server is open to CORBA objects.

However, the functions of the IDE's source code editor are much too basic and the lack of a server-side debugger will make development of large-scale projects quite difficult.

IDE productivity for server processing	Rating/10
Assistance with server-side code generation	6.3
Tools and automation provided by the IDE to improve productivity	4.2

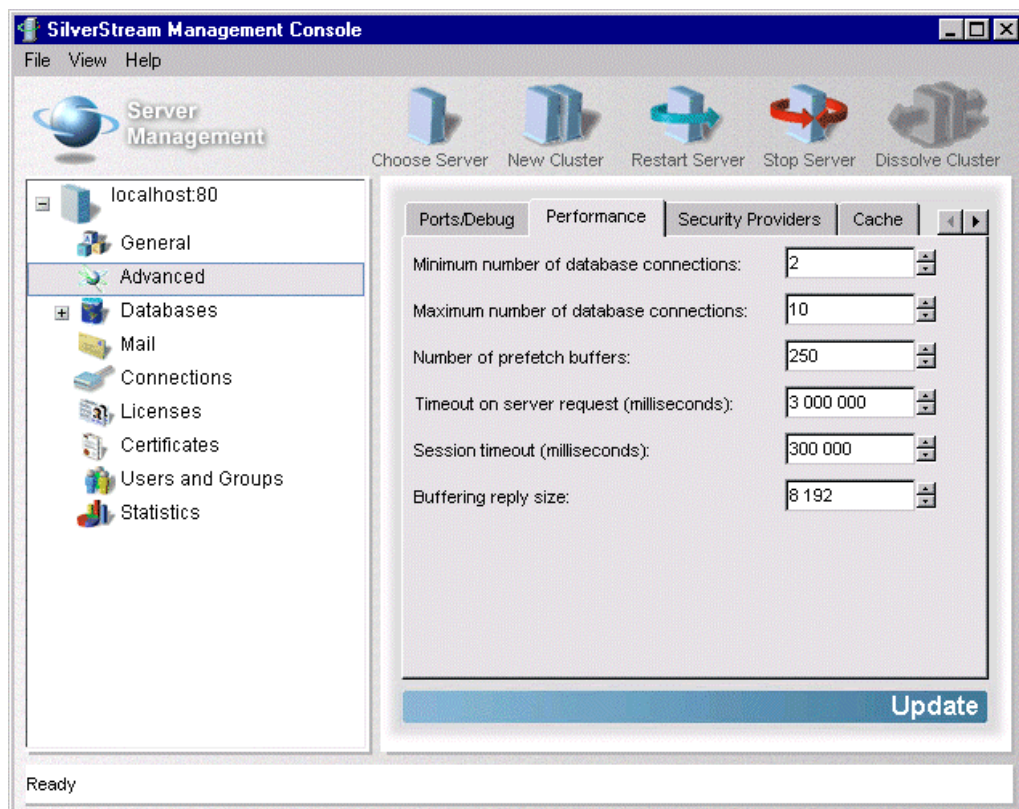
3.6. Application server

3.6.1. Database access

SilverStream bases database access on Sun's JDBC API. Several drivers ship with the product: Oracle, SQL Anywhere, Informix and DB2. The ODBC interface can also be used through the JDBC/ODBC bridge provided with the server. SilverStream teams redeveloped this bridge to make up for the shortcomings of Sun's model, no support for multithreading, for example.

SQL support is satisfactory as displaying/modifying BLOB-type (or long raw) fields is possible. Also, it recognizes DBMS-specific SQL queries such as MS-SQL Server's (`get date`).

In order to ensure good performance for database connections, SilverStream offers a JDBC connection pool for each database. However, there are some limitations to how one can configure the connection pool. Even though the minimum and maximum number of connections can be specified, it is not possible to specify maximum idle time or garbage collection interval. Be careful, the connection pool is only applicable on databases supported by SilverStream. However, it is possible to recover data from another database using a good JDBC driver, but the connection pool can no longer be used.



Configuration of JDBC pool in the SilverStream Management Console

In the case of DBMS failure and restart, SilverStream is not capable of restoring connections automatically. The editor has announced the addition of this function in the next version.

DBMS data is easily accessible from the HTML or Java interface generator. It is possible to link graphic controls to table fields or DSO columns.

3.6.2. Language richness, openness

By providing JDK 1.1 and basing its performance on Java, SilverStream provides developers with all the advantages of this language. All fundamental concepts of object programming are present: inheritance, polymorphism, abstraction, etc. Demanding developers will be pleased to find high-level APIs that cover such wide domains as the processing of character strings, mathematics, TCP sockets and security.

Variable management can be handled at session level or at application level. While dynamic page development is handled by Page Designer, context management is taken care of automatically. Should the developer choose the servlet API, he will have to manage the context with the help of Sun's standard APIs. This framework is rich because it allows for the invalidation of an HTTP session at any time. It also lets Java objects react if they find themselves bound to an HTTP session (HTTPBindingListener).

In order to facilitate connectivity to external systems, SilverStream provides DSOs. DSOs make it possible to access databases and Notes bases with the help of an ERP-unique API. The DSO API is open and it is possible to develop DSOs that access very particular sources such as real time throughput. Numerous DSOs are available separately and are ready-to-use for accessing SAP/R3, Baan Peoplesoft, Notes bases and the principal IBM systems (MQSeries, CICS...).

SilverStream's HTTP server supports SSL 3, in order to ensure data exchange confidentiality. It cannot be dissociated from the application server and it mandatorily treats requests. Use of a third-party front-end HTTP server is possible because SilverStream provides ISAPI and NSAPI interfaces. However, only the first HTTP request will be treated by the HTTP server; the following requests are rerouted directly to SilverStream. This must be taken into account when implementing network security.

3.6.3. Deployment

In order to guarantee a highly available service that is capable of supporting workload increases, SilverStream offers the possibility of clustering several servers.

Load balancing is based on three elements:

- The **cache manager** is a program responsible for synchronizing information on multiple servers that are found in the SilverMaster repository.
- The **load manager** updates the status of different servers that make up the cluster and periodically tests their accessibility. In this way, it establishes a map for rerouting requests.
- The **dispatcher** redirects requests according to the state of the servers, as indicated by the load manager.

The failover model is laid out particularly well. Two levels are proposed:

- **Server-level/application-level failover:** SilverMonitor is responsible for monitoring cluster components. Should one of these components fail, SilverMonitor restarts it. If failure is material, the server will be excluded from the cluster.
- **Session-level failover:** HTTP-session information is replicated. If a server fails and loses session information, it is transferred to another server.

3.6.4. Evaluation

➤ Summary

The connectivity of release 2.5 has been considerably improved. The main DBMSs, ERPs and IBM middleware are currently integrated. Java's functional richness, as well as cluster-mode architecture to maintain the load and ensure high availability, makes SilverStream a serious candidate for large-scale applications. Unfortunately, a third-party HTTP engine cannot be used and the number of server-supported platforms is limited.

Application server	Rating/10
Database access	8.1
Richness of language, openness	6.4
Deployment	6.4

4. Annexes

4.1. Workload methodology

➤ Introduction

Objective

To compare the performance and intrinsic behavior of intranet application servers, by testing (through the use of workload tests) the various implementations of TMBench 1.0 specifications presented here.

How?

- By relying on HTTP and SQL standards, rather than on products considered as references.
- By offering representative, independent intranet units instead of a complete integrated intranet application.
- By making the implementation of test units easier (simple and straightforward HTML layout).

Characteristics of the chosen intranet architecture

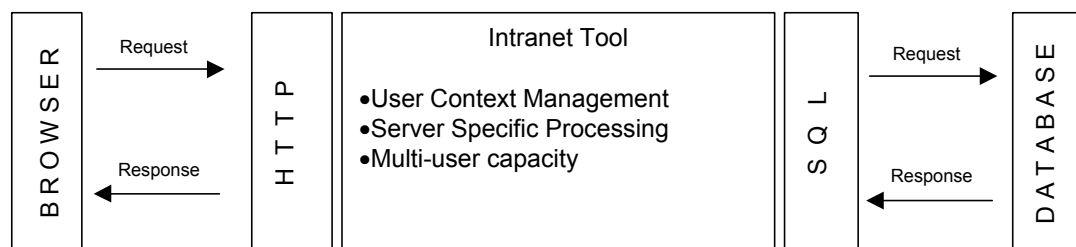
The application's clients are standard browsers (the application generates only HTML).

The development tools used to implement TMBench 1.0 must have the following intranet development features:

- The tool must have a programming or scripting language.
- The tool's application server must allow for algorithmic calculation and processing.
- The tool must allow access to the standard databases available on the market.
- User context management must be possible.
- Finally, the application developed by the tool must operate in multiple-user mode (for several concurrent users).

General principles of TMBench 1.0:

The tool specifications attempt to highlight the services provided by the intranet part by relying on two standards: HTTP and SQL.



A basic transaction is defined by a pair (request, response). The request is an HTTP request (the size and format of the URL are not specified). The expected HTTP response is entirely standardized (the result and format of the answer are imposed).

In requests involving database access, DBMS access is normalized with a SQL query. Most SQL queries cited in this document can be used as they are, but they can also be adapted to a specific intranet tool, mostly concerning the data access possibilities that the tool offers.

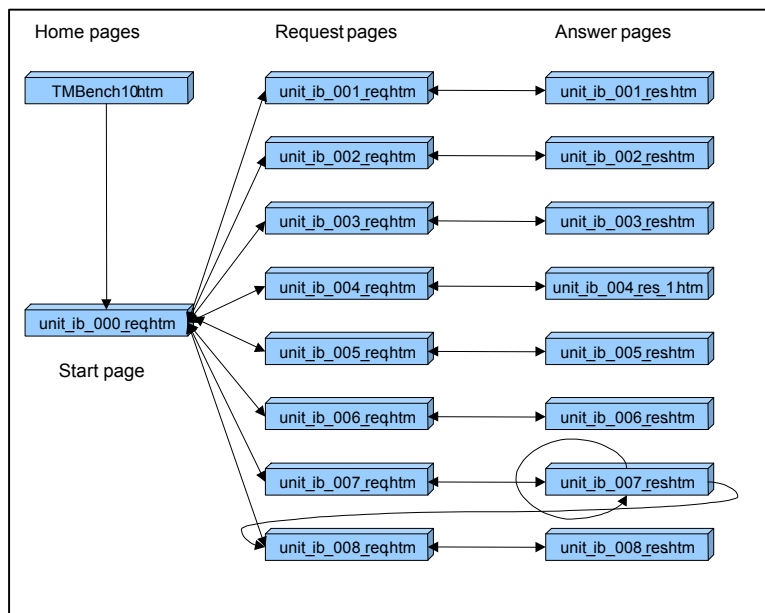
➤ **Database and SQL queries**

The database model used is of a relational type. No particular DBMS product is specified, and the choice of a database engine common to all TMBench 1.0 specifications is made at the last moment during the tests. The SQL queries must be as standard as possible (in other words, usable with the most popular DBMSs on the market).

The physical data model contains six tables, three relationships, 50 fields and a maximum of 60,000 records for a total data volume ranging from 10 to 100 MB. These contents may vary at any time in order to accommodate the specific needs of different tests.

➤ **Bench application**

The TMBench 1.0 application is made up of eight independent units, each including two pages (one request page and one response page). There is also a welcome page and a page that references the eight units. TMBench 1.0 is therefore made up of a total of 18 HTML pages. The application's kinematics is as follows:



TMBench 1.0 application kinematics (all of the HTML links)

Role of each module

The eight modules of the scenario allow us to center the analyses on basic features. The goal is thus to isolate each of the desired features and, using the dominant functional features of the desired application, to be able to anticipate the behavior of the deployed application in relation to the application server selected.

Modules	Role (basic feature analyzed)
Module 1	Large "Select" (+ than 5000 records returned) without cache in a database
Module 2	Sum of the small "Select" without cache in a database
Module 3	Sum of the small "Select" with cache in a database
Module 4	Multi-criterion search with dynamic request, without cache
Module 5	Database updates (simple insertion and updates)
Module 6	Algorithmic calculation
Module 7	Storage of user context in memory
Module 8	Mass insertion

The complete TMBench specifications can be downloaded from the TechMetrix Research Web site: <http://www.techmetrix.com/lab/tmbench/tmbenchindex.shtml>

Technical constraints

The editor must respect HTML page content as described herein as much as possible while keeping in mind the following constraints:

- HTML file names are given for example purposes only, you may choose others.
- At the top of each page there must be an HTML title and label (standard HTML text) indicating the test reference.
- A back to Start Page link must be found at the bottom of all "starting" HTML pages (requests).
- A back to IB-00X (previous page) link, which enables the request to be restarted quickly, must be found at the bottom of all HTML "answer" pages (request results).
- No JavaScript data input monitor is mandatory in the HTML pages.
- The request pages of IB-002 and IB-003 units possess "hard coded" SQL requests. These values can be modified at any time, thus making application evolution possible.
- The only freedom concerns URLs and their format (URLs of the HTML links and the SUBMIT buttons), as they cannot be imposed.

Format of normalized HTTP/HTML responses and requests

The format of HTML request and response pages must comply with the following guidelines as closely as possible:

- Resulting data is to be put in tables as soon as possible.
- The default table border must be equal to one.
- No specific format is to be applied to displayed data.

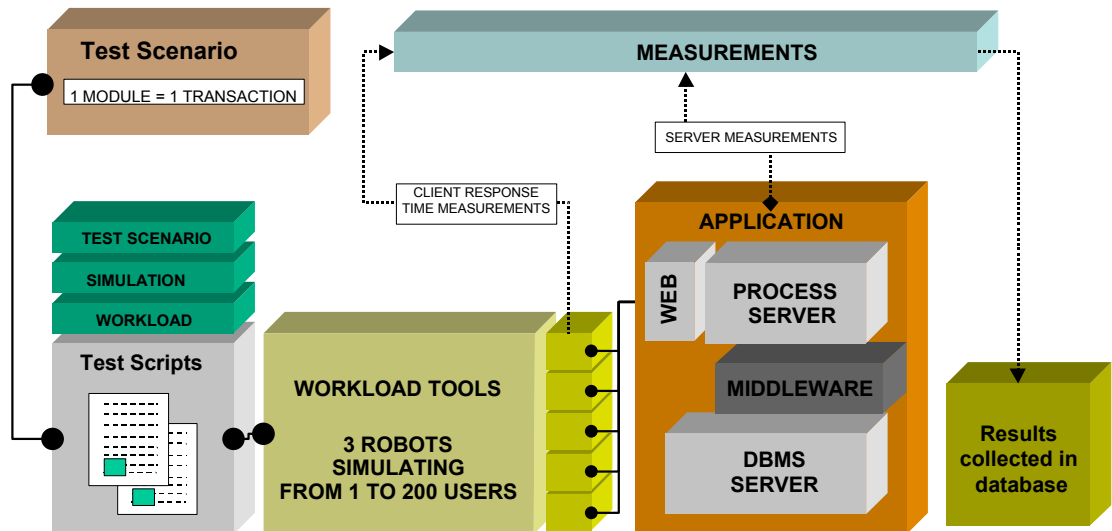
➤ Performance measurements

Once we have received and validated the integration complying with all of the TMBench 1.0 specifications, a workload test, simulating at least 200 concurrent users, is conducted. Breaking the application up into test modules makes it possible to define each unit as a test transaction independent from the others. Each transaction (called a scenario) is subject to a

measurement of client response times. When a unit is used several times in sequence, the whole of these iterations makes up a complete transaction.

Method used for workload tests:

Our method is based on pre-prepared test scenarios, which are conducted as shown in the diagram below. These tests, which involve the entire application architecture, are applied to each TMBench 1.0. implementation in the same way.



Workload test architecture

To have a better understanding of measurement analysis, note that each unit makes up a test transaction, and therefore a measurement in its own right. This measurement corresponds to a request and to the obtainment of its answer. The time required for a complete IB-001 transaction is equal to the sum of the amount of time taken to send a request and the amount of time taken to receive its result from the server. For unit IB-001, this measurement corresponds to the time needed to go from page *unit_ib_001_req.htm* to page *unit_ib_001_res.htm*.

Performance measurements and indicators

Measurements are made at three different levels:

- On the workload test robots, by collecting time values from HTTP server requests and answers with which the simulated users are interacting.
- On the processing server, with a measurement station which traces and logs the percentage of CPU time and the memory used, which expresses the load generated by the application server.
- On the DBMS server, by counting the number of connections opened by the application, and the load generated by the SQL orders.

4.2. Functional evaluation criteria

Each criterion is evaluated and then rated using one of the three following values:

- ☆☆ or YES: criterion supported correctly
- ☆ or YES Minus: criterion supported partially or is complicated to use
- ☹ or NO: criterion not supported or is much too complicated to use

4.2.1. Productivity of the development environment

➤ Quality and richness of the development environment

Step 1: Installation, ease of learning, simplicity

N°	Description	Rating	Comments
1	Quality of integration between tools (respect of unique window).	☆☆	During development, passage between different tools (Page Designer, Object Designer, Form Designer...) is transparent. These tools have unified design and ergonomics.
2	Printed documentation and online help with search options.	☆☆	Documentation is complete and readable. It includes an administration manual, a programming manual and a tutorial. Online help is also provided with a search engine.
3	Ease-of-use and installation. Installation mustn't call for too many prerequisites (browser, DBMS, HTTP server). The developer must be able to use the wizards quickly.	☆	For the most part, installation is well assisted and well documented. However, installing SQL Anywhere requires tricky configuration and three reboots between the installation of Fulcrum, SQL Anywhere and SilverStream. Numerous wizards are then available.
4	Richness and quality of the tutorials and examples provided. The examples must cover all internet-related problems (scripting language, DBMS access, context management...).	☆	The tutorial explains the basic functions of the product. Advanced features (CORBA, DSO, Cluster...) are explained in the programming manual. However, it lacks some sample applications to illustrate all server functions (IOP and DCOM).

Step 2: Completeness of the product, openness to other tools

N°	Description	Rating	Comments
1	Completeness of the product in terms of development, deployment, administration.	☆☆☆	The product is complete. Tools cover development and deployment with Designer and SilverStream Management Console.
2	Presence of a project and project resource manager, quality of the interface with PVCS, VSS, etc.(check-in/check-out, versioning).	☆	SilverStream allows for management of application resource rights. No development version management mechanism is provided. However, there are interfaces with PVCS, MS VisualSourceSafe 5 and 6. Interfacing with a versioning tool provides SilverStream developers with new icons that make check-in/check-out possible.
3	Tool for developing multilingual applications (listed in a dictionary, post-development extraction).	●	
4	Test HTTP server and a local development DBMS.	☆☆☆	An HTTP server is integrated into the application server and the DBMS, SQL Anywhere, is provided.
5	Presence of report-generating tool, the possibility of interfacing with a report generating tool (for DOC, RTF, HTML, or PDF printing). The reports editor must allow for control breaks and offer a test mode.	☆	There is no native, report-generating tool. However, SilverStream integrates such a tool in the form of a servlet. This tool, Jreport, is from the start-up company Jinfonet Software. Reports can be read using applets in an HTML page, JavaBeans in an application or in HTML page form (Jreports for SilverStream is currently available in beta version).
6	Presence of a search engine and/or quality of the interface with a search engine.	☆☆☆	Fulcrum Search Engine ships with SilverStream. It can be used with all SilverStream-developed applications.
7	Presence of a relational and/or object modeling tool, possibility of interfacing with Power AMC, Rational Rose, Mega, etc.	☆	Table Designer allows relational modeling (table creation, relationship specification). No object-modeling tool is provided.
8	SQL query editor, with support for stored procedures (run and display source code). The editor must allow for query input and provide, in real time, without passing by the application server, the result of this query. Richness of the authorized SQL would be a differentiating factor.	●	There is no SQL query editor.
9	Tree structure and application display management tool. From a page, the tool must propose all of the called pages and the request pages. Any change in the page name must be monitored or taken into consideration by the calls. A graphical view of the pages and/or window is a plus.	●	Page linking is not displayed graphically.
10	Repository, cross object references (graphics, components, objects used by the IDE...), detection of unused objects, export of standard objects, etc. For example, during the deletion of a "database connection" object, the IDE must inform the developer of the different uses of this object.	●	Repository objects do not have cross-references between them: Deletion of images or DSOs takes place without doing the same for their references in the repository.

➤ HTML interface generation

Step 3: Automation of HTML interface generation

N°	Description	Rating	Comments
1	Presence of a WYSIWYG HTML editor. The developer does not need to know HTML in order to design his pages.	☆☆	HTML interface construction is done graphically with the help of Page Designer; knowledge of HTML development is not required.
2	Numerous table manipulation possibilities (cell mergers, cell breakdown, nested tables)	☆☆	Page Designer makes it possible to define tables, merge and split cells.
3	Presence of a repository for HTML graphic components (objects or groups of objects).	●☆☆	It is impossible to create complex HTML components and archive them in a repository. It is only possible to integrate JavaBean components in a palette and to use them in the IDE.
4	Possibility of creating style sheets, templates, and page models.	●☆☆	The HTML code generated by SilverStream Designer is 3.2-compatible. It is possible to integrate style sheets in developments, but no assistance is given; corresponding code must be entered manually.
5	HTML code generation during the insertion of Java or ActiveX components and images in HTML pages.	☆☆	Insertion of Java components, ActiveX components and images is accompanied by configuration of their behavior in an HTML page.
6	Client-side generation of validation scripts.	●☆☆	SilverStream does not allow automatic generation of client-side validation scripts. It is however possible to build server-side validation expressions.

Step 4: HTML interface development: assistance and potential

N°	Description	Rating	Comments
1	Presence of an HTML display tool and/or an HTML source editor with syntax recognition.	☆☆	HTML editing is done according to two modes: visual composition and source code editing. The HTML-source-code editor facilitates readability by syntax recognition and coloring.
2	Presence of dynamic HTML page generation wizards.	☆☆	Wizards allow for the generation of dynamic HTML insertion pages, and to update/modify database records. Pages can then be modified with Page Designer.
3	Import/export of HTML pages and total command of HTML (static and dynamic, in other words, generating HTML code by programming).	☆☆	Dynamic HTML pages are developed visually and the code generated can be accessed and modified. The tool is bi-directional. The only way to import HTML code remains, however, copy and paste.
4	Assistance with HTML input (tag list, completion, documentation...) and client-side scripting language (events, objects, authorized attributes and methods, documentation...)	☆	No HTML input assistance is provided; only syntax coloring validates the code entered. With regard to ECMAScript entry, Designer lists page-object events, but does not help with code input.
5	Presence of client-side scripting language components and Java applets.	●☆☆	Neither scripting language nor Java applet components are provided.
6	HTML 4.0, DHTML and DOM are taken into account.	●☆☆	DOM is supported, but no assistance is offered. SilverStream generates neither HTML 4 nor DHTML code; only HTML 3.2 is supported.

➤ **Java graphical interface generation**

Step 5: Elaboration of the Java graphical interface: automation and assistance

N°	Description	Rating	Comments
1	Presence of a Java WYSIWYG editor.	☆☆	
2	Presence of a components palette (AWT, JFC...).	☆	A Java-AWT graphics component palette is offered. Some proprietary components enhance this palette. SilverStream has announced that Swing components will be supported in the next version.
3	Possibility of creating JavaBean components.	●☼	Creating JavaBean components with edition of properties, methods and events and BeanInfo class generation is not assisted. However, it is possible to create Java archives (JAR) and include all types of files within these archives (classes, GIF and JPEG images...) in order to facilitate application deployment. The manifest file is generated automatically.
4	Possibility of creating Java applets.	☆☆	
5	Possibility of enhancing the Java graphics object palette and the presence of an object inspector.	☆☆	It is possible to integrate JavaBeans into a project. They are accessible in the Media palette, which is available during development.
6	Presence of dynamic transactional application generation wizards with a Java graphical interface.	☆☆	Wizards make it possible to create applets or applications for updating table records.

➤ **IDE Productivity for server processing**

Step 6: Assistance with server-side code generation

N°	Description	Rating	Comments
1	Management of user profiles with application and page access security.	☆☆	Application access security can be configured for development and operation at page-level. The security manager can connect to NT directories.
2	Configurable wizards for creating transactional applications, with the possibility of modifying code after it has been generated.	☆	Configurable wizards for creating transactional applications are available and their code can be modified. However, their configuration is not possible.
3	Assistance in creating standard distributed objects (ActiveX, CORBA, and EJB).	☆	SilverStream supports CORBA objects via VisiBroker ORB. ActiveX components are supported inasmuch as Microsoft's virtual machine is used; however, no assistance is offered for developing them. SilverStream has announced EJB support in the next release.
4	Server-side assistance in sending e-mail (high-level API, examples, documentation).	☆☆	SilverStream provides classes for creating, sending or reading e-mail (SMTP/POP 3). The documentation dedicates an entire chapter to this subject; diverse points such as treating attached documents are explained with examples. IMAP 4 is not supported.
5	Automatic management of the unique session identifier.	☆☆	Context management is ensured entirely by cookies; it is handled automatically by a client-side session identifier.
6	Freedom to choose the context management used for transfer of the identifier (URL longs, cookies, and hidden variables).	●☼	For context management, the use of cookies is imposed.
7	Freedom to choose between a client-side and server-side stored context, with the possibility of tracing history.	●☼	No session history management. An identifier is managed client-side, but the context is memorized server-side.
8	Assistance in File Uploading (interface, code examples, explanations).	☆☆	

Step 7: Tools and automatic functions provided by the IDE to improve productivity

N°	Description	Rating	Comments
1	Good quality source code editor, with find and replace features, context-specific help, completion, syntax parser, etc.	★	A source code editor is available. Syntax coloring, completion, and find/replace are available. However, multiple-file searches are not supported.
2	Complete debugger (logs, step by step, choice of the application or tracing in functions/methods/procedures, modification of variable content).	●	A complete debugger for the client parts of Java applications is provided (step-by-step, stopping points, method tracing). Variable modification is not possible in debugger mode. There is not a server-side debugger.
3	Assistance with mapping between business objects and the application interface.	★	SilverStream supports technical objects, but does not have a framework that allows true object modeling. It is thus impossible to bind the graphical interface to business objects. However, SilverStream is based on the relational model. The Relational Data Palette allows binding database fields to application graphic components very easily.
4	Presence of an application management console.	★★	SilverStream Management Console allows for application administration. Cluster and security configuration is possible, as is consultation of server statistics.
5	Presence of an application use/consumption administration tool with graphics and optimization solutions.	★	It is possible to see connected users and active Java threads. Also, one can observe the number of HTTP requests received and their response time. However, no graphics are provided.
6	Presence of a testing tool and/or workload tool.	●	

4.2.2. Application server richness

Step 8: Database access

N°	Description	Rating	Comments
1	Support for ODBC and/or JDBC (type 1) driver.	☆☆	The JDBC/ODBC bridge was rewritten by SilverStream teams to improve efficiency (multithreading support is not provided by the Sun driver).
2	Native database access (SQL Net, Dblib-CTLib, Inet) via type 2 and/or type 4 JDBC.	☆☆	SilverStream ships with the following drivers: - JDBC (type 2) driver for Oracle - JDBC/ODBC bridge Drivers from other editors are also provided: - JDBC (type 4) driver for Ingres - JDBC (type 4) driver for Sybase - JDBC (type 4) driver for Informix - JDBC (type 4) driver for DB2
3	Support for BLOBs/Long Raw (reading and insertion).	☆☆	
4	Support for DBMS-specific SQL queries (Ex: select getdate (), select banner from v\$version...).	☆☆	
5	Support for stored procedures, with the transfer of in/out parameters.	☆☆	
6	Support for failure and reconnections to the DBMS. If the DBMS is stopped while there is an existing connection pool, the application will take care of restoring connections itself. At least a programming solution is possible.	☹	DBMS connection restart after failure is not supported. SilverStream has announced a system of connection restart for version 3.
7	Support for multiple data sources within one page or graphical window. Possibility of creating standalone connections.	☆☆	Within an application it is possible to create several DSOs that access diverse sources. The fields of these DSOs can be used within a same page or applet. Standalone connection creation is possible by creating a JDBC connection manually and being very careful to close the connection after using it. Creating shared connections means a connection pool must be used.
8	Extended management of connection pools (multi-instances and maximum number of connections).	☆	It is possible to specify the maximum and minimum number of database connections, but detailed pool management is not possible. One cannot specify the maximum length of a connection, the maximum idle time or the garbage collection interval.

Step 9: Language richness, openness

N°	Description	Rating	Comments
1	Creation of functional classes that support object polymorphism.	☆☆	Java is totally integrated into SilverStream as development language. Java classes support polymorphism and can be organized in packages inside the server.
2	Support for all variable types (dates, multi-dimensional tables, dynamic tables, and structures).	☆☆	Java supports a large number of types (int, char, double...) and just as many tables with n dimensions. When creating classes, one creates his own types.
3	Creation of features, with transfer of variables and recursive elements.	☆☆	Java supports methods.
4	Variable management at session and application level.	☆☆	SilverStream offers the possibility of managing session and application variables.
5	Presence of features/methods which define the length of time-out (at session level, dynamically, and at application level for all of the sessions).	☛	Management of session-level timeout is not possible. SilverStream has announced that this function will be available in version 3.0.
6	Presence of complete invalidation management features/methods (possibility to force the end of a session and launch a process when an event "log-out" is received).	☆	It is possible to invalidate using the HttpSession method invalidate (). However, it is not possible to trigger an event on log-out.
7	Server-side generation of print files using values coming from the data source.	☆	Using the third-party tool, Jinfonet Jreports for SilverStream (available in beta version) makes it possible to generate PDF or CSV files or to print them.
8	Server-side generation of image files (GIF, standard formats) using values coming from the data source.	☛	
9	Capacity to handle File Upload (the tool must allow for the recovery of HTTP flow in a string for example).	☆☆	
10	Presence of an object/relational mapping module.	☛	RDBMS-data mapping on business objects is not supported by SilverStream.
11	Presence of a distributed objects system.	☆☆	SilverStream is interoperable with VisiBroker CORBA 2 ORB.
12	Possibility of interfacing with a standard distributed object model (DCOM, CORBA, and EJB).	☆	SilverStream proposes interfaces with CORBA objects via VisiBroker. It is also possible to create IDL files with Designer. EJBs will be supported in version 3. With regard to DCOM objects, SilverStream supports them inasmuch as the Microsoft JVM is used. There is no assistance or integration in the designer.
13	SMTP, POP3/IMAP4 and LDAP support.	☆☆	SilverStream supports e-mail send and receive. LDAP directories can be used to configure security.
14	The processing server is an HTTP and FTP client.	☆	Since it supports JDK 1.1, SilverStream provides the HttpURLConnection class (which inherits from URLConnection), which allows connection to a Web page by its URL and recovery of the content via OutputStream. However, there are not any classes for manipulating FTP flow.
15	Possibility of proposing different levels of security for applications (SSL, SET, use of firewalls...).	☆	Implementation of the SSL V3 protocol in the server guarantees integrity between Web clients and the SilverStream server. The server does not offer SSL connection between Java clients and the server. It is impossible to limit SSL encryption to a single page.
16	Openness to ERP, with "business" modules adapted to ERP (SAP, Peoplesoft...).	☆☆	DSOs mask the complexity of accessing the main ERPs on the market. DSO (EDC) generators are adapted to Peoplesoft and SAP R/3. However, these EDCs are often sold separately.
17	Openness to TP monitors (Tuxedo, Encina...), integration middleware (MQ Series...).	☆☆	DSOs mask the complexity of accessing the main TP monitors and market-asynchronous middleware. There are DSO (EDC) generators that are adapted to CICS, Tuxedo and MQ Series. However, these EDCs are sold separately.
18	Consideration of XML (XML application server, reuse of XML components...).	☛	XML is not taken into account.

Step 10: Deployment

N°	Description	Rating	Comments
1	Multi-platform application server: Unix, Linux, and Windows NT.	★	The deployment platforms supported are Windows NT 4, HP-UX 11, and Solaris 2.5. The development platforms supported are Windows 95/98/NT4.
2	Support for several types of interface with HTTP servers: CGI, ISAPI, and NSAPI.	●☼	SilverStream can only be used with its own HTTP server. There are however gateways which reroute HTTP requests from the Web server to the SilverStream server. Netscape and Microsoft Web servers are supported. Be careful, not all HTTP requests pass through the front-end Web server. In fact, only the first request reaches it, the following requests are rerouted to SilverStream. This point must be taken into account when considering network security.
3	Possibility of deployment on other application/object servers (ASP, LiveWire, NCA cartridge, EJB, JSP, servlets...).	●☼	SilverStream applications cannot be deployed on other environments. However, if the code respects Servlet 2 API to the letter, applications can be deployed on other application servers. In this case, no advantage is taken from SilverStream in terms of development.
4	Support for application-level and session-level failover.	★★	SilverStream supports cluster mode and proposes failover at two levels: Application-level failover: SilverMonitor guarantees good performance of cluster components and provokes their restart should the program fail. If the component suffers material failure, the load-balancing tool excludes it from the cluster. Session-level failover: The cache manager ensures replication of HTTP-session data of cluster servers. This avoids losing HTTP sessions that are managed by a server that fails.
5	Possibility of modifying the application without stopping the services ("on the fly" modifications).	★★	Modifying code does not necessitate restart. Changes can very well be made "on the fly".
6	Possibility of spreading the server processes across several physical machines (distributed objects created with the tool, or relying on the application server), and of separating the HTTP server, the application server and the DBMS server.	★★	It is possible to spread server treatments across several machines that run the same application. It is possible to disassociate the database server. However, the HTTP server cannot be dissociated from the processing server.
7	Support for load balancing, with the possibility of configuring the balancing algorithm.	★★	Load balancing is possible on several SilverStream servers. Customization of the load-balancing algorithm will be supported in the next version. However, it is possible to balance loads on servers following constants.

A study conducted by
TechMetrix Research

Authors

Nicolas Farges
Franck Gonzales

Translation and Editing

Gina Faucher

Contributors

Emmanuel Gourion
Christophe Lauer
Benoît Léger
Philippe Mougin

Publication date: November 1999

USA

TechMetrix Research
6 New England Executive
Park, Suite 400
Burlington, MA 01803

Tel. : 1 781-270-7486
Fax : 1 781-270-7487

<http://www.techmetrix.com>
info@techmetrix.com

EUROPEAN HEADQUARTERS

SQLI
55/57 Rue Saint Roch
75001 PARIS
FRANCE

Tel. : (33) 01 44 55 40 00
Fax : (33) 01 44 55 40 01

<http://www.sqli.com>
r&d@sqli.com

SWITZERLAND

SQLI
World Trade Center
Avenue Gratta-Paille 2
CH-1000 LAUSANNE 30
PO Box 476

Tel. : (41) 021 641 10 65
Fax : (41) 021 641 13 10

<http://www.sqli.ch>

WARNING: Intellectual Property Act.

Art. L 335 - 2: All publication of written works, musical compositions, paintings or any other literary output, printed or engraved, in full or in part, in defiance of the laws and regulations relative to the property of authors, constitutes forgery, and all forgery is a misdemeanor.

Forgery in France of works published in France or abroad is punishable by two years imprisonment and a fine of 1,000,000 F (L. no. 94-102, 5 Feb. 1994, art. 1st)

Art. L 335 - 8: Legal entities may be held legally responsible under the provisions of article 121 - 2 of the Penal Code for infractions defined under articles L 335-2 to L 335-4 of this act.