TechMetrix research

Product Review
HahtSite 4.0

A report by
TechMetrix Research

TABLE OF CONTENTS

1. Product profile

Identification	
Product composition	HahtSite IDE and Application Server
Release	4.0
DBMSs supported	Oracle, Sybase, Informix, MS/SQL Server, all ODBC and JDBC
Development platforms	HP UX, Solaris Sparc, AIX, NT 4
Deployment platforms	Same as development platform
Editor	Haht Software

Haht Software is an American company created in 1995 by the founders of Q+E Software. Their product, HahtSite, is a complete tool for transactional Web application development and deployment. Thanks to a development environment with an integrated HTML WYSIWYG graphics editor and an application server, HahtSite makes for good productivity right from the beginning of the learning phase.

Simplicity and comprehensiveness were the two main strengths of previous releases of HahtSite. With a script-oriented application server, a development tool that was both rich and intuitive, HahtSite seemed to be one of the market leaders in this category. Since then however, the product has been beefed up quite a bit and HahtSite is now playing in the big leagues. The first step in this transition was allowing server-side load balancing and thus no longer limiting processing to one physical machine. Then, the arrival of Java as a potential development language facilitated openness to existing information systems. Although release 3.1 only offered a hybrid solution based on HahtTalk (version adapted from Microsoft VBA), release 4 leaves the choice between HahtTalk scripting language and Java object language entirely up to the developer.

At the same time, HahtSite benefited from other important improvements, notably concerning the application server. For example, session failover support guarantees great availability for the deployed applications. Other elements, presented as complementary modules, make interfacing with Tuxedo, accessing CICS mainframes and reusing SAP applications possible. It is unfortunate that most of these interfaces, called ESMs, are presented as components to be downloaded from the Haht Software Web site. This factor prevents HahtSite from being a true packaged product. On the other hand, if we look at the interface offered on the client station, we can see that HahtSite is entirely dedicated to HTML interface applications.

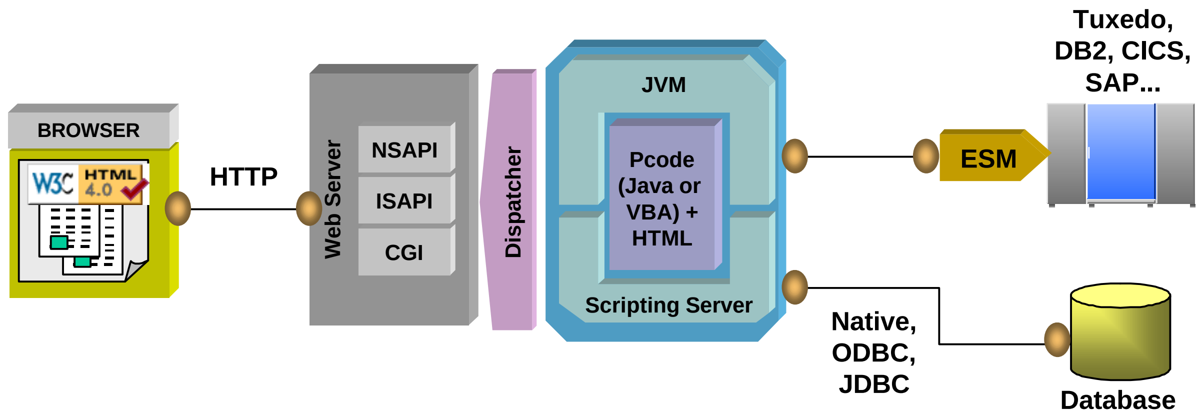
The richness in terms of deployment and development can be seen in the functional evaluation conducted for this report. With the exception of Java interface generation, a problem it is not designed to handle, HahtSite received above-average marks in all of the evaluated areas. In terms of productivity, HahtSite's greatest strength remains its HTML page design workshop. This workshop makes graphically creating high-quality HTML dynamic pages possible while allowing for the import/export of these pages from your favorite design tool. The development environment can be characterized by its simplicity. However, it still has some shortcomings in terms of its integration with other tools (modeling tool, project management, etc.), which keeps it from being a true AGL. Finally, to improve productivity, HahtSite provides relatively complete tools (debugger, source editor, etc.) which are a definite plus when creating specific processes.

The application server also obtained good results. As it was one of the weak points in past releases the editor made considerable efforts to help HahtSite move up a notch. Whether it be database access, language richness with high-level generic methods or its deployment potential, HahtSite has certainly

made a lot of progress. The only lack is in terms of its openness to existing elements, which is for the most part due to the need to use ESMs that are not integrated into the packaged product.
Regarding performance, HahtSite, with an application entirely developed with Java, supported perfectly a workload of 200 concurrent robots. On neither HTTP request or database transaction level did an error occur. The response times recorded during the most sustained tests were good; while with a single-user configuration they were mediocre for the chosen physical configuration. But the most important point remains the application's behavior during heavy use and in this context, HahtSite proves to be very good. This problem comes mostly from the deployment configuration, where with a database connection per user, the limitations could be seen during real-world business deployment (the number of database-authorized licenses is rapidly exhausted if several applications access the same RDBMS).

In conclusion, HahtSite is without a doubt, from a technical point of view, one of the most mature and complete environments available in the area of Intranet application servers. Offering most of the basic services necessary to implement complex applications, thanks in particular to the application server's completeness, HahtSite can today be considered as a viable business solution. In addition, its complementarity with its editor's development tool makes it a complete product that covers the entire development cycle, from the design stage to the administration phase. The biggest weakness of HahtSite concerns its durability as the deployed applications could be in the balance should the company be purchased.

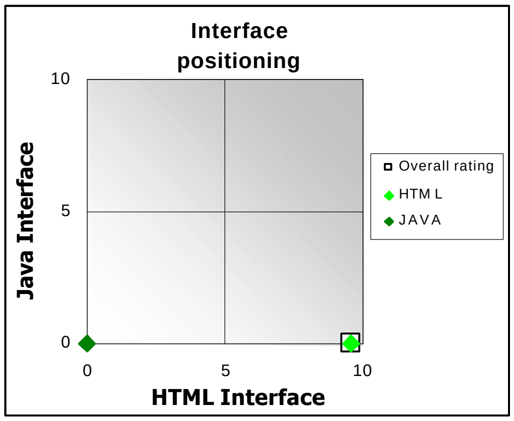
Architecture

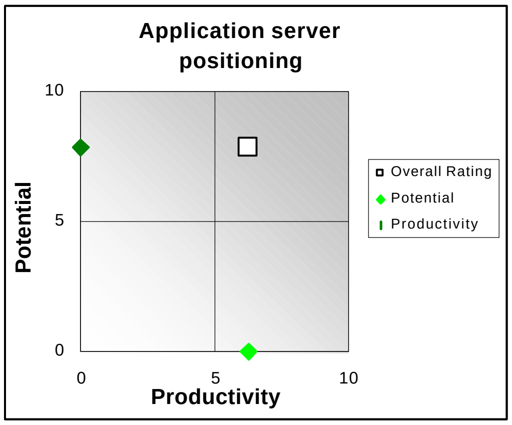


HahtSite respects the idea of universal client. An HTML interface is offered during deployment, while the proprietary modules remain server-side. Here, the freedom to choose between the house VBA and Java allows the tool to adapt to different profiles and different functional needs. HahtSite makes it possible to spread different application processes across different physical servers and opens itself up to the information system of a business thanks to ESMs.

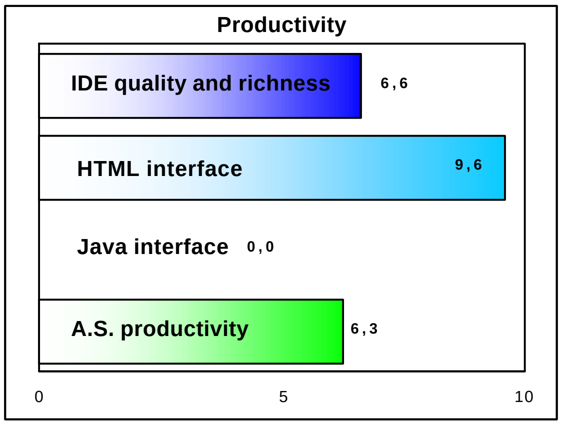
The Pros	The Cons
▸ Help in creating an HTML interface	▸ Lacks API richness
▸ Ease of learning and use	▸ No object modeling tool
▸ Failover strength	▸ Not open to Java graphical interfaces
▸ Presence of two languages	

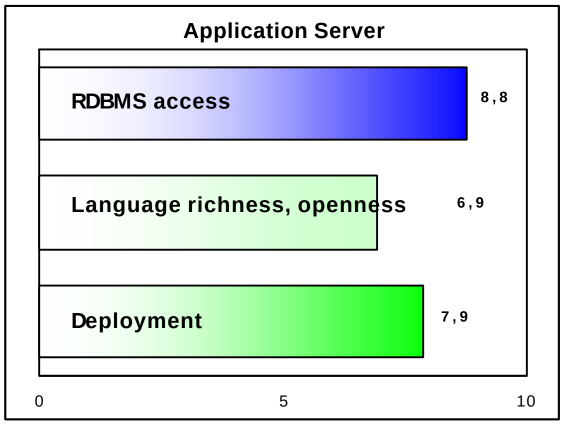
Functional Evaluation





As we can clearly see in the first positioning graph, HahtSite is dedicated to HTML intranet application elaboration. With a rating of 0, Java interface is not really the tool’s strong point. However, HahtSite’s results in terms of HTML interface generation and assistance make it one of the best on the market. In regards to server-side possibilities, HahtSite release 4.0 offers even more than its forerunners. It is the addition of Java as an alternative to HahtTalk Basic in its latest release that will allow HahtSite to attack a bigger market as a serious candidate for large enterprise projects. Its average rating of 8 out of 10 in this area proves this point. Beyond the application server’s potential, it is important to note the development tool’s productivity, as it makes quickly putting complex applications into place even easier.





Completeness is one of HahtSite's best qualities. In terms of IDE, the developer has on hand many assets to be productive quickly. The only thing missing is a greater openness to complementary tools (search engines, modeling tools, etc.) to broaden the development environment’s overall potential. Furthermore, with the exception of Java interface generation for which HahtSite is not designed, the product provides good productivity.

The application server can be characterized by the richness of its programming languages (Java and HahtTalk) and good openness to other RDBMSs. But HahtSite’s biggest force resides in its deployment potential, with high application availability via the possibility of session-level failover. This strong characteristic of top-of-the-line environments is thus installed in a historically script-type environment which allows to best reconcile simplicity and performance. The only lack is in terms of API richness and openness, as low-level, fastidious programming is often necessary to meet specific needs.

Performances

180

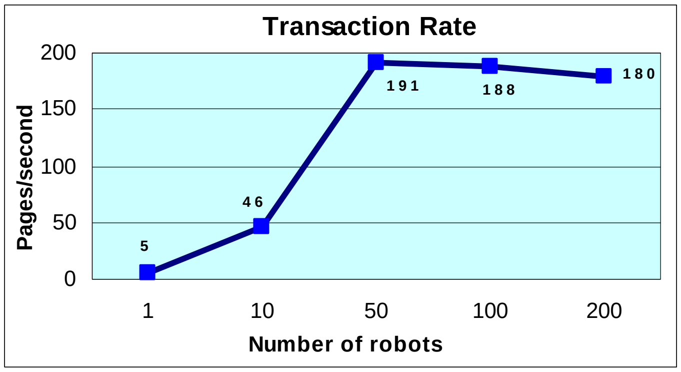
This is the number of dynamic HTML pages delivered per second by the HahtSite application server during the running of a “mixed” scenario (stringing together of 8 modules) with concurrent use of the application by 200 robots.

The HahtSite application was deployed on the platform without any difficulty. The first tests showed the application server’s solidity, as it didn’t show the slightest sign of weakness even with the maximum workload. No error interfered with this stage of the tests.

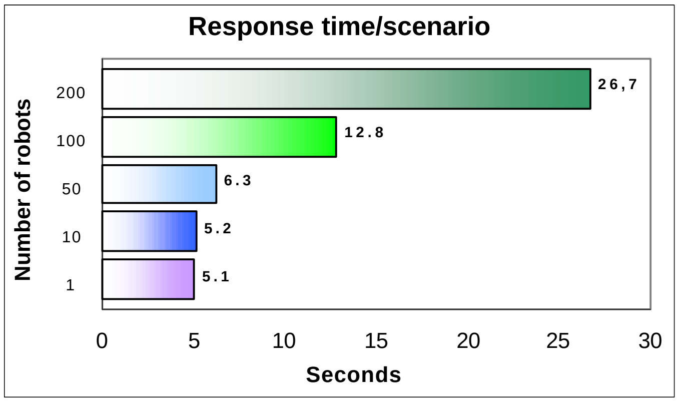
RELIABILITY 100%	**HTTP Errors : 0**
	DBMS Errors : 0

HahtSite’s behavior during workload tests reveals the reliability and consistency of the application server.

The drop in server-side performance is only slight (191 and 180 pages/second during the transition from 50 to 200 robots), which means that heavy use of the server does not result in a debasement of behavior.



With a single-user configuration, HahtSite takes 5.1 seconds to deliver 24 HTML pages, which is not particularly quick considering the material configuration. With 10 robots, application servers and data are better used because there is hardly any deterioration from the user’s point of view. This is also true with 50 concurrent robots. Then, the deterioration of response time is in proportion to the increasing number of robots.



Technical profile

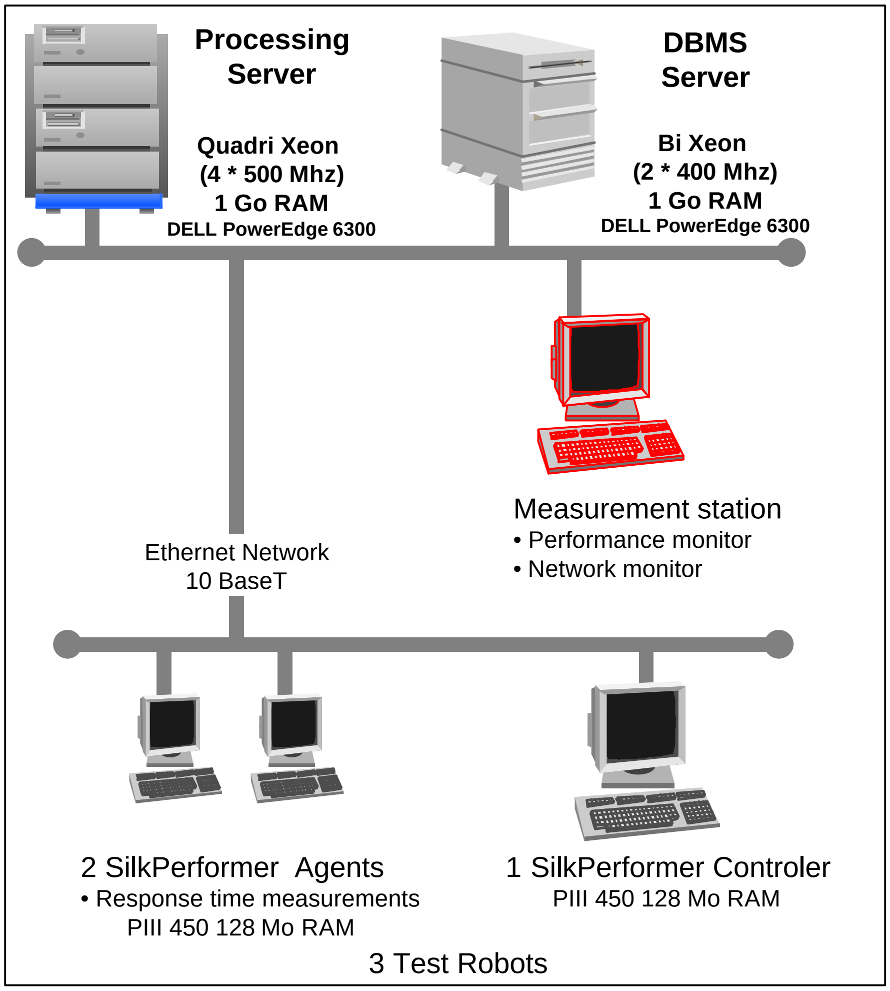
Development license	Approximately $2,000 per developer
Deployment license	Approximately $7,500 for 25 connected users (then from, on average, $150 per additional user depending on the bracket, tapering charges)
Configuration used for evaluation	HahtSite 4.0 for Windows NT 4.0 workstation
Complementary modules	ESM NLS (multilingual), ESM for Crystal Report, ESM for Adobe Acrobat

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 :**

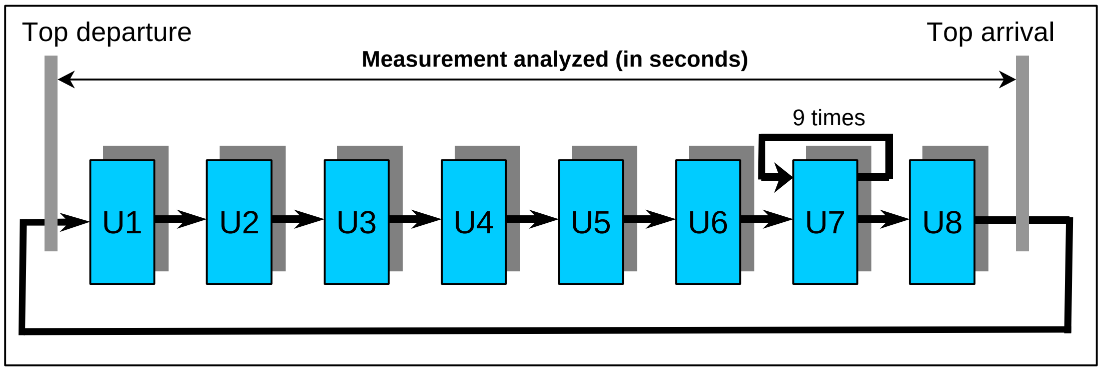
The application is broken up into eight distinct modules (or units), each of which represents one of the main needs of transactional intranet applications. Each model is characterized by its focus on a specific intranet need (simple search, update, calculations, etc.) while remaining as simple as possible 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 (stringing together 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, etc.) 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 module 4 only 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, as it indicates whether or not 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 flow)**

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 flow: the flow 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 the editor implements to get the most out of his application server. These are interesting points that can be used to judge the application server’s possibilities 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 application was entirely made with Java. Development complies, for the most part, with traditional HahtSite development. Only the "DataAgents", IDE objects that facilitate the link between form data and the database request results were not kept.

➢ **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%**

The workload increases went off without a hitch with HahtSite. Whether it be in terms of deployment, the running of the application or the application's reliability with maximum workloads, no problem was encountered. With 200 concurrent robots, which represent roughly 1000 concurrent users in a real-world configuration, we can seriously talk about the good reliability of HahtSite's application server for an application with a high rate of usage.

➢ **Analysis of user-side results**

Average	1 robot	10 robots	50 robots	100 robots	200 robots
Response time per 24-page scenario (in seconds)	**5.1**	**5.2**	**6.3**	**12.8**	**26.7**
Response time per page (in seconds)	**0.2**	**0.2**	**0.3**	**0.5**	**1.1**

We notice that with one, 10 or even 50 concurrent robots, the response times are practically the same. This highlights the relative slowness of the application server in a single-user configuration. The response times with a single robot are around 5 seconds for completion of a scenario (stringing together of 8 modules, or 24 dynamic pages), which is rather slow considering the physical configuration used.

With 50 robots, the times are practically identical, which is more than satisfactory at this level. After that, during the transition to 100 and then 200 robots, we run into a bottleneck that results in an increase in response time in direct proportion to the growing number of robots.

This consistency is extremely important as it shows the tool's capacity of supporting heavy use in a satisfactory manner.

➢ **Analysis of server-side results**

Average	1 robot	10 robots	50 robots	100 robots	200 robots
Transaction rate (pages delivered per second)	5	46	191	188	180
Transaction flow (time in seconds between delivery of two pages)	0.212	0.022	0.005	0.005	0.006

With 200 robots, the rate of dynamic HTML pages delivered by the HahtSite application server is roughly one page every 6 thousandth of a second. Even without using additional servers dedicated to HTTP server management or complementary modules, this already represents such a configuration in a particularly sustained usage situation.

From 50 concurrent users on, the transaction rate remains more or less stable (from 191 to 180 pages per second, or 7.9 to 7.5 complete scenarios), which goes to show the strength of the application server. In fact, it is with a single-user configuration that HahtSite has the most trouble using the physical configuration available.

2.2.2. General characteristics

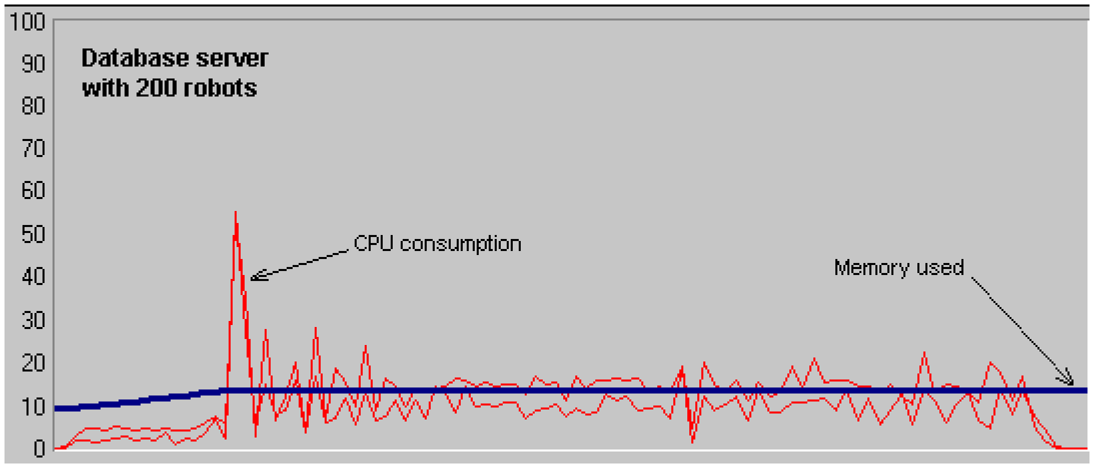
Configuration	1 robot	10 robots	50 robots	100 robots	200 robots
Number of open connections to Oracle database	1	10	50	100	200
Number of instances (or processes) started	8	8	8	8	8
Number of threads per instance	1	1	1	1	1

The deployment configuration chosen for the application is very simple. Indeed, the number of database connections is directly linked to the number of application users, while the number of launched instances (or processes) and threads per instance remains steady throughout the tests.

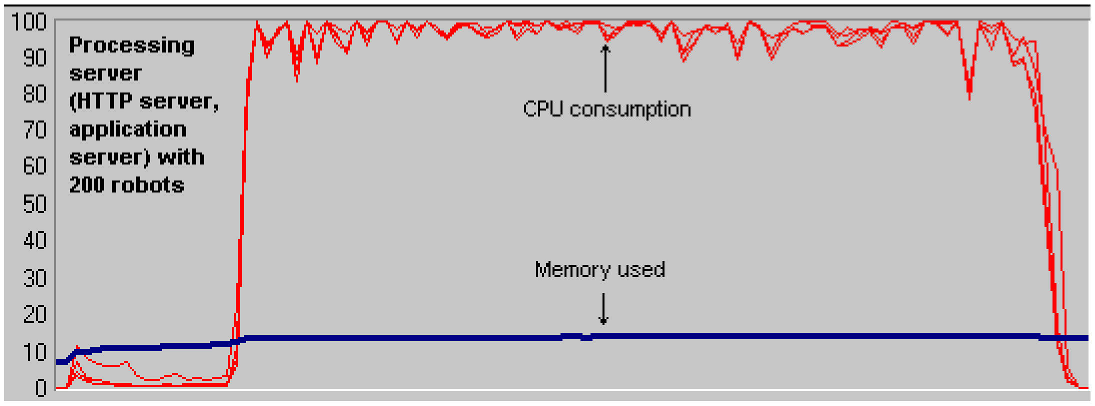
For database connections, this solution presents some disadvantages with regard to the use of a connection pool that makes limiting the slots to a RDBMS possible. Systematic connection can indeed pose problems in terms of the amount of server-side memory used or the blocking of some users if the maximum number of connections has been reached. Even if connection pooling is supported in HahtSite release 4, the editor opted for the systematic connection solution.

In terms of launched instances, the small number (eight) saves server-side memory and leaves the machine available for other tasks. The other advantage concerns the ease of the application's administration, as it need not be configured in relation to the number of users present at a given instant. The configuration tested on our platform addresses most issues and can thus satisfy applications destined to accommodate approximately one to 1000 (or even more) concurrent users.

➢ **CPU and memory consumption with 200 robots**



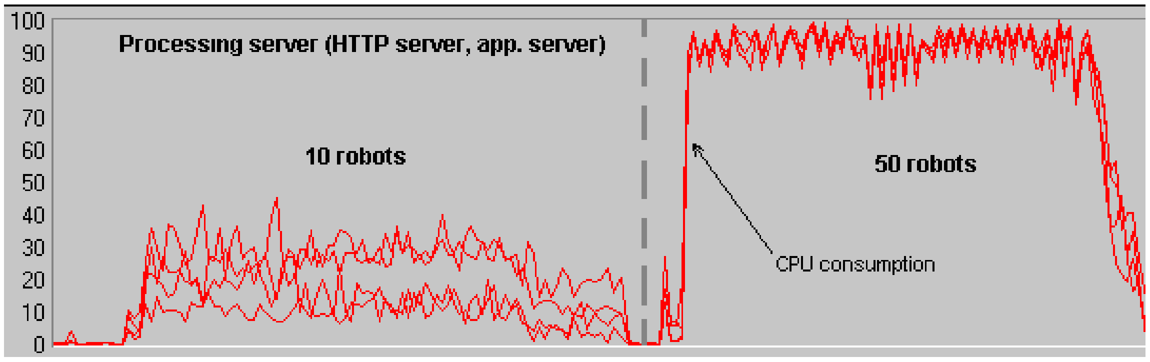
*CPU consumption (2*400 MHz) and memory used (1 Go of RAM) on the database server (in %)*



*CPU consumption (4*500 MHz) and memory used (1 Go of RAM) on the processing server (in %)*

The above graphs clearly show at which workload level the bottleneck occurs. With 200 concurrent users, it is the processing server's quadri processor (HTTP server and application server) that is saturated. The four Xeon processors (P3 at 500 MHz) also find themselves practically 100% occupied. Moreover, only 150 MB of RAM are required on this server. On the database, the two Xeon processors (frequency of 400 MHz) are only 10-15% engaged, while 150 MB are also required. It should be noted that at this level, the use of an Oracle 8 database significantly reduces consumption, as an Oracle 7 database already took up 700 Kb * 200 connections ⇒ 140 MB for the database connections alone.

- **CPU consumption with 10 then 50 robots on the processing server**



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

This graph highlights from which point on the bottleneck appears. With 10 robots, processing server's four processors are nearly 70% available. However, with 50 concurrent robots, the limit of maximum use is already reached.

This thus explains the results obtained and confirms why the response times worsen starting with 50 robots and then practically double. It is indeed from this point on that distributing processes between other physical machines should be considered in order to guarantee equivalent performance.

2.2.3. Table of results

➢ **Average time, in seconds, for each module of the "mixed" scenario**

	Modules							
Number of robots	1	2	3	4	5	6	7	8
1 robot	0.24	0.24	0.24	0.41	0.39	0.62	2.24	0.66
10 robots	0.63	0.24	0.27	0.41	0.42	0.65	1.85	0.68
50 robots	0.43	0.38	0.33	0.44	0.54	0.94	2.26	0.94
100 robots	1.48	0.92	0.65	0.68	0.97	1.76	4.59	1.68
200 robots	4.09	2	1.36	1.4	2.01	3.43	9.15	3.24

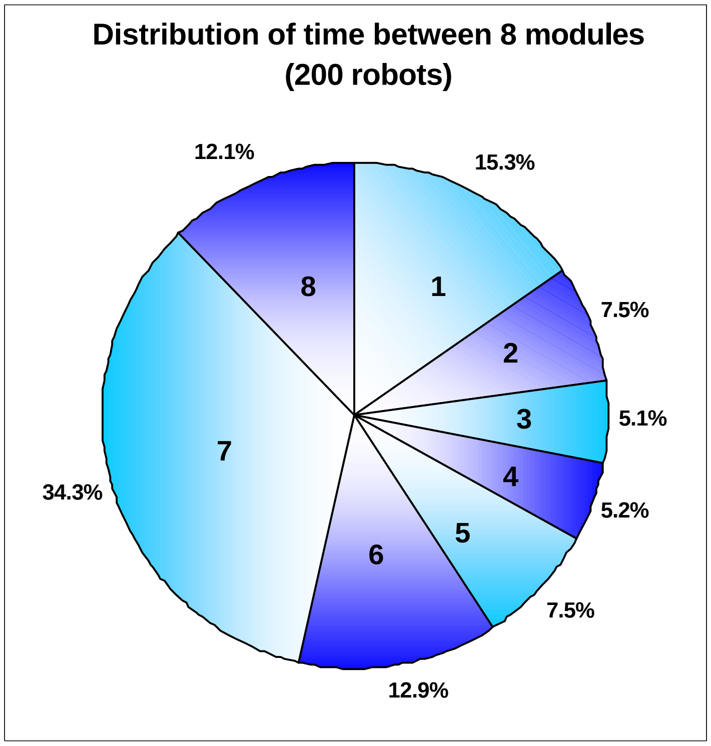
The average times obtained per module logically go up with the number of robots. We found nonetheless two situations in which this is not the case. Firstly, the mean time to complete module 1 is longer with 10 robots (0.63 second) than with 50 (0.43 second). This can be explained by the latency periods caused by the other robots that fluctuate in relation to robot use, thus shifting idle time. Module 7 was completed more quickly with 10 robots (1.85 seconds) than with one robot (2.24 seconds). In this case, as we will see below, the explanation lies more with the reuse of session-related objects.

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

	Modules		
Number of robots	4	5	7
1 robot	0.41	0.39	2.24
10 robots	0.41	0.4	1.97
50 robots	0.58	0.24	0.97

In analyzing these measurement details, we can see that larger the number of robots, better the response time for module 7. Given that there are 8 instances of launched applications per measurement, increasing the number of robots results in the reuse of session objects by several robots, which means a reduction in the average response times. However, only extremely in-depth analyses would allow us to explain HahtSite's behavior during module 5 (insertions and data updates), during which better response times are obtained with 50 automates than with 10.

➢ Distribution of time per module



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

Module 7 uses the most time during the tests. However, the nine passages of this module (designed to inflate the context for a mass insertion in module 8) generate 10 dynamic HTML pages (module initialization, followed by 9 result pages). Thus, if we divide the time obtained by 5 (2 pages for the other modules), module 7 of context management is not greedier than the other modules.

In fact, it is therefore module 1 (large select in the database) and module 6 (algorithm) which take the most time. However, the distribution is relatively consistent and HahtSite does not present any weak point in particular.

3. Functional evaluation

3.1. Introduction

Created in 1995 by the founders of Q+E, Haht Software, the Raleigh, NC-based company offers HahtSite, an environment dedicated to the development and implementation of intranet applications. The release discussed in this study is HahtSite 4.0, on the market since February 1, 1999.

3.2. Development environment quality and richness

3.2.1. Installation

➢ **Product description**

In this release, HahtSite includes:

- a development tool or IDE (Integrated Development Environment)
- an application server reserved for tests
- the HTTP server, Apache 1.3.4

We will come back to the installation of a distributed application server in the chapter devoted to deployment.

➢ **Prerequisites**

The development tool can be installed on either Windows 95/98 or Windows NT 4.0. The PC must have at least a Pentium 90 with 32 MB of RAM and 60 MB of free space on the hard drive.

It is recommended to have a browser and possibly an HTTP server should you wish to use something other than Apache 1.3.4 that is provided with the HahtSite product.

It is equally recommended to have a pre-installed JDK, especially if you wish to develop with Java. Nonetheless, these products can be installed later, but will thus necessitate configuration of the IDE and the application server.

➢ Procedure

The installation program ("InstallShield") begins by updating the JVM, if necessary, by installing Microsoft's version 5.00.3165.

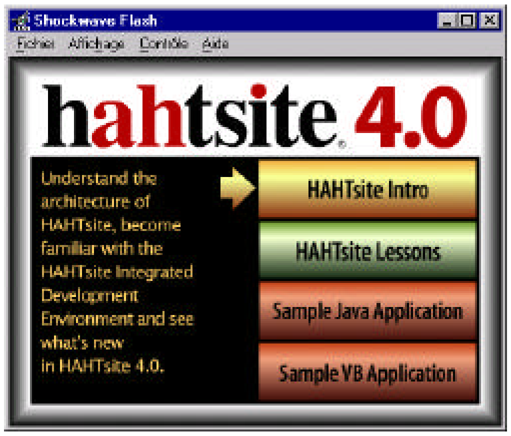
The «complete distribution » option begins by installing Apache 1.3.4, and then the IDE by selecting the default browser that will be used to display pages (Netscape or Internet Explorer).

Finally, comes the installation of an application server dedicated to testing with automatic detection of previously-installed HTTP servers and publication site configuration.

➢ Documentation and tutorials

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

- for the IDE : installation manual, programming manual, user's manual and a Widget programming manual
- for the application server: installation manual and an administration manual, which is also available in PDF or by download from the Haht Software Web site (www.haht.com)



Understand the architecture of HAHTsite, become familiar with the HAHTsite Integrated Development Environment and see what's new in HAHTsite 4.0.

HahtSite Tutorial

A complete tutorial, in "Shockwave" format, provides a presentation of the product and a complete lesson in HTML format. The tutorial is made up of 11 lessons which become progressively more difficult as well as two applications (VB and Java) which group together a large part of tool's features.

After a few hours, the user succeeds in creating a small application and gets a good idea of the tool's features.

3.2.2. Composition of the product

➢ **A complete environment**

The development tool offers a good level of integration, grouping together the following on the same IDE:

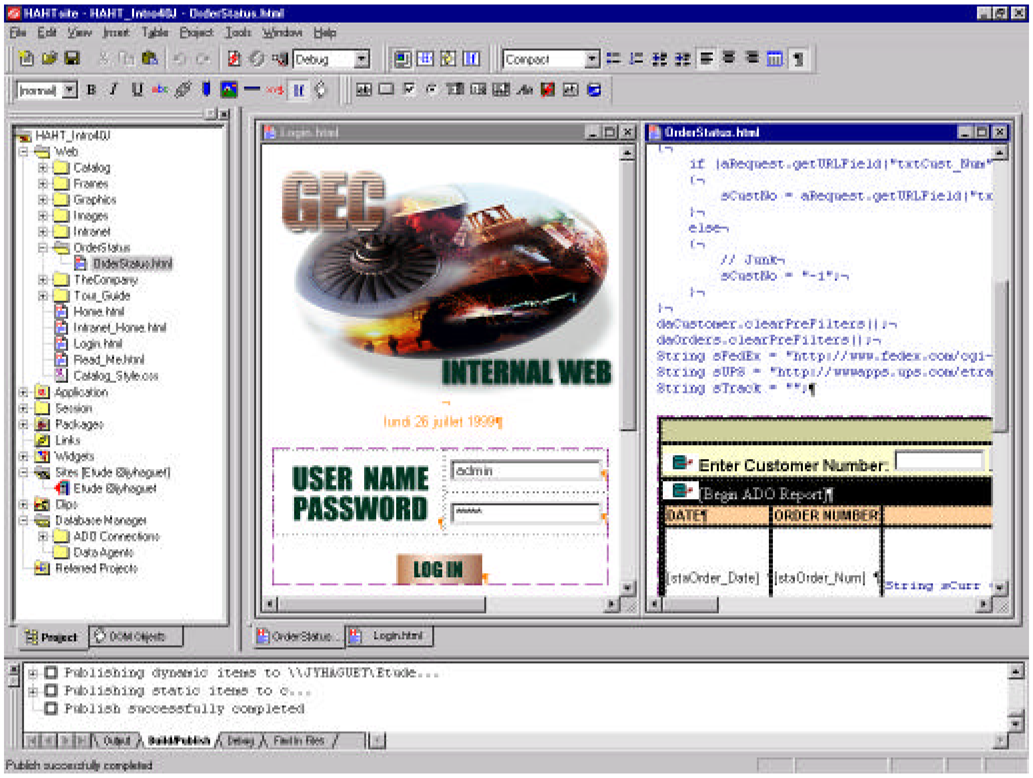
- HTML pages (static or dynamic)
- images (GIF, JPEG or PNG)
- style sheets (CSS1)
- client script files (JavaScript and/or VBScript)
- processing files (Java and/or VB)
- reusable HTML components («Clips »)
- publication sites
- database connections
- Widgets (server-side, code-generating components)

All of these objects are compiled on the project explorer; it is possible to create repertories to regroup components by generic theme.

Depending on the project component, the editor plays the role of:

- an HTML WYSIWYG editor
- an HTML tag editor
- an image editor
- a client script editor
- server-side source code editor (VB or Java)

Only the image map editor is an external application, but it has a good level of integration with the IDE.



```
{
    if (aRequest.getURLField("txtCust_Num"
    {
        sCustNo = aRequest.getURLField("tx
    }
    else
    {
        // Junk
        sCustNo = "-1";
    }
}
daCustomer.clearPreFilters();
daOrders.clearPreFilters();
String sFedEx = "http://www.fedex.com/cgi-
String sUPS = "http://wwwapps.ups.com/etra
String sTrack = "";
String sCurr
```

```
Publishing dynamic items to \\JYHAGUET\Etude...
Publishing static items to c...
Publish successfully completed
```

The IDE allows for the creation of HTML pages and server-side source code (VB and/or Java)

- **Openness to third-party tools**

HahtSite does not provide developers with profile features, work group or versioning tools. However, it enables interfacing, with a good level of integration, with Microsoft Visual Source Safe but also with Merant Intersolv PVCS, StarBase StarTeam and MKS Source Integrity.

To create multilingual applications, HahtSite offers the use of ESM (Enterprise Solution Module) NLS 3.0 (National Language support), which is downloadable from the Haht Web site. This tool is based on a dictionary that can be enhanced with an editor. The translation language is deduced in relation to the browser configuration or it can be imposed during page loading.

HahtSite does not possess an HTTP-specific server. The application server is HTTP/1.1 client and thus recognizes most of the Web servers on the market: Apache, Microsoft, Netscape, Oracle Web Request Broker, etc.

To create reports, HahtSite offers "ESM for Seagate Crystal Reports" which allows for easy integration of reports coming from Seagate Crystal Reports 5.0 or higher. Another ESM makes generating dynamic reports possible. This offers the possibility of browsing all of the data in Adobe Acrobat (.PDF) and even modifying it provided the user have Adobe Acrobat Exchange.

➢ **Incomplete building blocks**

With regard to data modeling, whether it be Merise, UML or other, HahtSite does not possess tools and does not offer an interface with another market tool such as Power AMC, Rational Rose, Mega, etc. This clear-cut separation between modeling and creation is attributed to the business object aspect that one can expect from a tool based on an object language such as Java.

The development tool does not possess a true SQL editor to run SQL queries or to create stored procedures. In fact, there is a trick that allows us to go through a shared query, but the ergonomics of such a solution is questionable. In any case, if the application structure is based on stored procedures, a creation tool, modification and deployment of stored procedures are indispensable.

If the application requires a search engine, it is necessary to turn to the complementary HTTP server or to another product on the market.

3.2.3. Evaluation

➢ **Learning process**

The product's installation (IDE and development application server as well as definition of a publication site) should not take more than half a day, even for a novice.

Thanks to tutorials with a thorough and convivial interface, learning to use the development tool is not difficult.

Those having experience with the previous release (3.1) are not destabilized; general ergonomics is more or less the same and the new features integrate without any unpleasant surprises.

➢ **Summary**

The development environment provides a good balance between functional richness, cohesion between the elements that make it up and simplicity of implementation.

The only hitches that demanding developers will find concern the lack of modeling tools (object or relational), openness to other market tools, and an exhaustive dictionary that would have the role of establishing cross references between all objects of an application (variables, pages, routines, etc.) and even establish technical documentation.

IDE quality and richness	Rating/10
Installation, learning process, simplicity	8.8
Completeness of the product, openness to other tools	4.5

The ease of installation and learning as well as the simplicity and ergonomics make for a promising first contact with the client.

The product's completeness has some shortcomings, notably its lack of a dictionary; also some modules such as the report tool (under a form other than HTML) and multilingual support are available as ESMs.

ESM designates all the modules that Haht Software provides by download from its Web site (http://www.haht.com) and which fill the roles of some of the development tool's missing features. The interest of such a solution is that each person can enhance the standard product with the modules that interest him. Each module is designed with a sufficiently modular logic and offers a number of parameters. However not all ESMs offer the same level of integration with the development environment and the "hot line" is not always guaranteed.

There are ESMs in different areas:

- report editing to create dynamic forms in PDF format (in Adobe Acrobat)
- e-commerce application access security
- LDAP feature support (single sign-on, workflow, etc)
- openness to Notes
- openness to mainframes by Shadow Direct (Neon Systems)
- multilingual support with NLS (National Language Support)
- openness to SAP R/3
- integration of reports in Seagate Crystal Reports format

3.3. HTML interface generation

3.3.1. HTML interface design and optimization

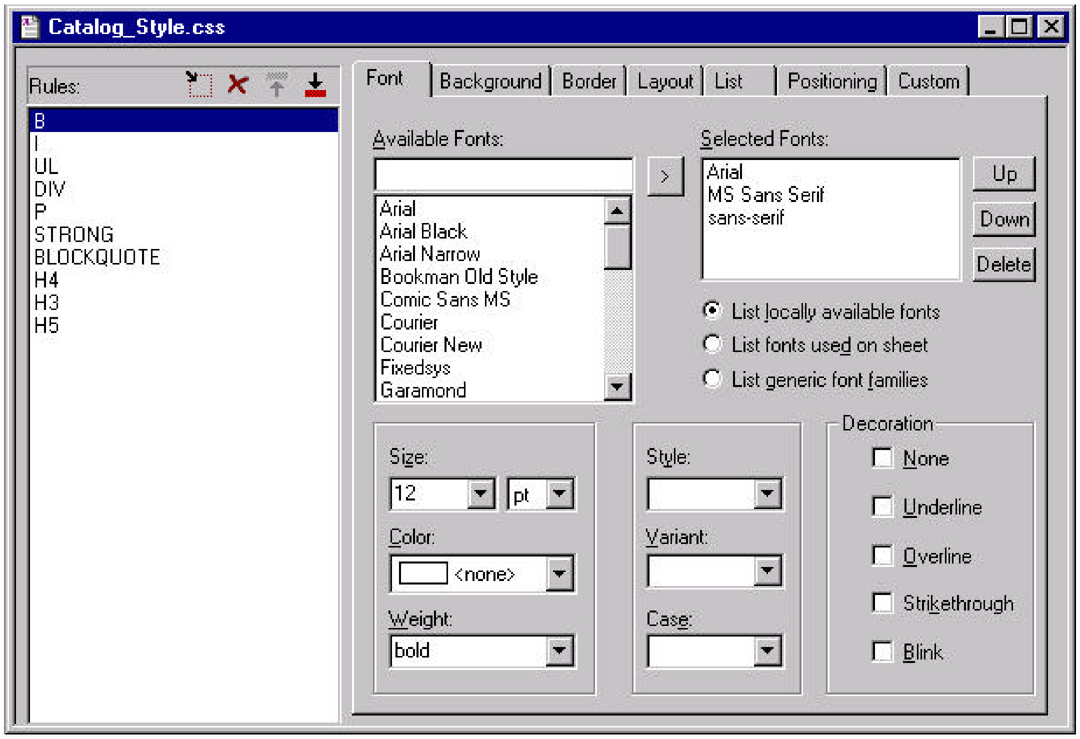
➢ **HTML page creation**

The HTML page editor provides a WYSIWYG mode. Page creation is made considerably easier by the presence of a palette containing the standard form fields (static text, text box, button, checkbox, etc.) and a dialogue box allowing for definition of their properties.

HahtSite provides page creation wizards (connection, master/detail, form, and list); they are configurable and can be modified after their generation. One can complete the list by adding one's own page templates; the page template is not part of the project, but its availability depends on the path configured in the IDE options.

To easily put headers and footers in place (or any other bit of HTML code that is recurrent on all pages) the best solution is to define "Clips" and place them on HTML pages. Any change to a clip is made to all pages containing it.

Style Sheets are another feature destined to optimize and homogenize HTML interface creation. HahtSite fully supports W3C's CSS1 recommendation (and CSS2 partially) and offers an editor for creating style sheets.



Rules: B, I, UL, DIV, P, STRONG, BLOCKQUOTE, H4, H3, H5

Available Fonts: Arial, Arial Black, Arial Narrow, Bookman Old Style, Comic Sans MS, Courier, Courier New, Fixedsys, Garamond

Selected Fonts: Arial, MS Sans Serif, sans-serif

[✓ List locally available fonts] List fonts used on sheet / List generic font families

Size: 12 pt
Color: <none>
Weight: bold
Style:
Variant:
Case:

Decoration: ☐ None ☐ Underline ☐ Overline ☐ Strikethrough ☐ Blink

Style sheets creation window

An import feature makes integrating a page or a repertory into a project possible. All pages, even those imported, can be displayed in tag form. The editor thus offers syntax coloring and validates the HTML code entered. In this way, one keeps total control over the HTML. The list of HTML tags and the associated values can be accessed from the form field properties.

Table creation is incredibly simple. It is possible to include a table in a cell and merge cells horizontally and vertically.

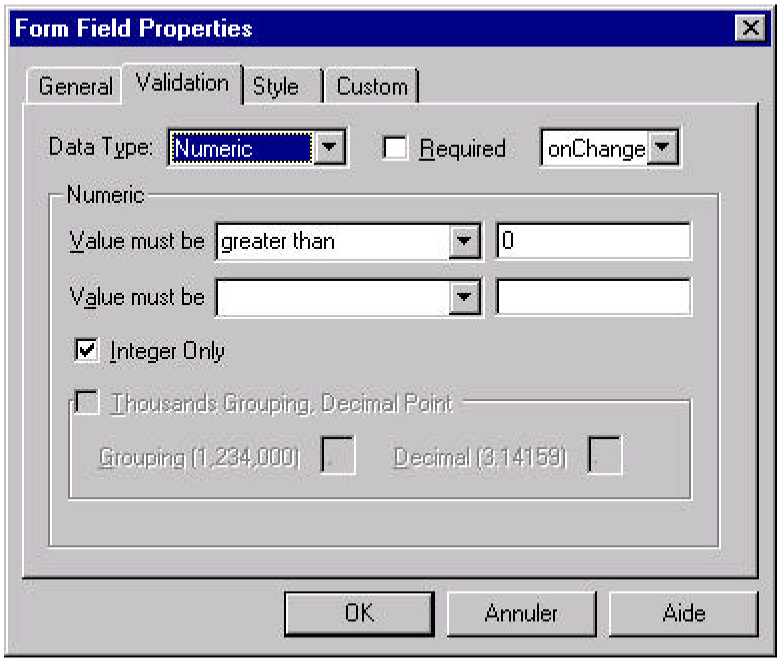
The image editor recognizes GIF, JPEG, and PNG formats. Even though an image map editor is not integrated into the IDE, it allows for references to be made to project components.

A spell checker allows for the verification of a selected page or section of text.

Finally, if despite all the features offered you wish to insert a Java applet or a multimedia component (LiveAudio, LiveVideo, QuickTime, RealAudio, Shockwave), HahtSite offers a wizard that helps you configure the object. For an ActiveX object, it is necessary to use the Widget with the same name.

➢ Client-side processing

In addition to the HTML properties of a form field, it is possible to define the field validation criteria.



Data Type: Numeric
☐ Required
onChange

Numeric
Value must be greater than 0
Value must be
☑ Integer Only
☐ Thousands Grouping, Decimal Point
Grouping (1,234,000) Decimal (3.14159)

Dialogue box allowing for the generation of input control JavaScript code

The associated JavaScript features are thus generated within the HTML page.

Note: it is possible to personalize the generated JavaScript source code by modifying the files that are found in the repertory HAHTSiteIDEpath\scripts\FieldVal (to send messages back in a specific language for example).

HahtSite provides help with client script input by recovering the page's DOM (Document Object Model) as a hierarchical tree structure: the client script explorer. Here all of the possible page and present field attributes, methods and events can be found, with an indication regarding their compatibility with the main browsers on the market (Netscape Navigator 2.0 and 3.0, Netscape Communicator 4.0 and Internet Explorer 3.0 and 4.0) for each of them.

The client script explorer possesses a "client-side" node that groups together all of the JavaScript features defined for the page by the developer.

It is possible to create client script files within the same file. The features contained in this file will be accessible from any HTML page or script file.

The client script can also be coded in VBScript. However, HahtSite will only generate JavaScript. It is recommended not to use VBScript except in the case of an intranet application for which one is 100% sure that the browser target is Internet Explorer.

Finally, the client script can be present directly on an HTML page in WYSIWYG mode. Then all one has to do is select it and "mark" it as client script: it then appears in a different color. This manner of coding on the HTML page in WYSIWYG mode can be used for relatively simple pages. However, it shows its limits with more complex pages, especially if the HTML page is dynamic, in other words, if the page contains server-side code (see section 3.5, IDE productivity for server processing).

3.3.2. Evaluation

The automation of HTML interface generation is close to perfection. The presence of wizards, the possibility to make page templates, to define clips, the CSS1 style sheet support, and nested table management allows for quick and easy generation of a good quality HTML interface.

The flexibility and the freedom to choose client script as well as automatic generation of input controls and the presentation of the page DOM make for good assistance in creating a thorough and carefully prepared graphical interface.

HTML interface generation	Rating/10
Automation of HTML interface generation	**10**
Assistance and potential of HTML interface creation	**9.2**

It is difficult to find a weak point in the HTML interface automation. The possibility of personalizing wizards would have made the editor impeccable.

With regard to the assistance and potential of HTML interface creation, the only things really missing are additional Java and ActiveX examples and a more complete HTML tag editor (completion, tag list, truly contextual assistance).

3.4. Java graphical interface generation

3.4.1. Creation of the Java graphical interface

Concerning the graphical interface, HahtSite does not offer an alternative: it will be done in HTML or it will be done with another tool.

If you want a tool which allows you to use Java for application and applet creation, HahtSite is not the tool for you as it is difficult to imagine such type of development without a Java WYSIWYG editor, without a components palette (AWT, JFC, Swing, JBCL, etc), without an objects inspector, without the possibility of creating JavaBeans or without wizards.

Of course, HahtSite allows for applet creation, but offers no assistance whatsoever. The developer would have to code everything.

3.4.2. Evaluation

No need to beat around the bush, with HahtSite creating applets is only possible if you are a Java expert as no assistance is provided.

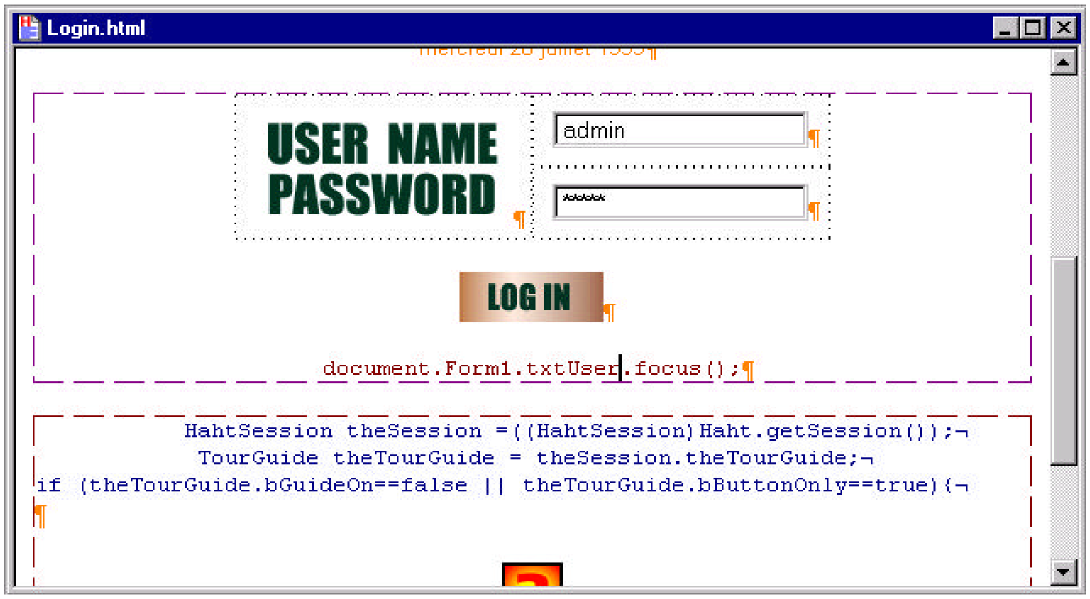
Java interface generation	Rating/10
Automation and assistance in elaborating Java graphical interfaces	**0**

3.5. IDE productivity for server processing

3.5.1. Assistance in generating server-side code

With release 4.0 HahtSite adapts even more to Java. Already in release 3.x, it was possible to include Java classes (source or byte-code) in a project. Now, HahtSite offers the choice between HahtTalk (which is largely based on Microsoft VBA) and Java as the programming language.

As soon as the project is created, you must choose between HahtTalk (with the possibility of including Java code) and a project entirely in Java. No matter the choice, the editor remains the same for the HTML interface and the server code. In this way, on an HTML page in WYSIWYG mode, it is possible to make the distinction between free form text, HTML form fields, client script code in JavaScript and server-side code in HahtTalk or Java.



```
document.Form1.txtUser.focus();¶
```

```
            HahtSession theSession =((HahtSession)Haht.getSession());¬
             TourGuide theTourGuide = theSession.theTourGuide;¬
if (theTourGuide.bGuideOn==false || theTourGuide.bButtonOnly==true){¬
¶
```

Display showing both the HTML page and the server-run code

The Privileges intermediary can handle user profile management. Once defined, privileges can be attributed to different pages. It then suffices to indicate the current privilege level at session level.

Some wizards make creating page types such as identification pages, or master/detail pages possible. HahtSite then generates the corresponding server code and allows for the addition of other processes.

With a concern for modularity, it is possible to create reusable modules called Widgets. Some are included in the standard product, and a wizard and a volume of the printed documentation are invaluable sources of assistance in creating new ones.

HahtSite allows for the creation of CORBA objects by defining their IDL, but no assistance is provided. It is however impossible to create ActiveX components or JavaBeans.

Sending and receiving e-mail is very easy. The objects “smtp” and “pop3” complete this task once the properties such as the gateway, the addressee and the sender are filled. It is also possible to attach a document.

Context management is the application server’s responsibility. It is situated on the same level as two fundamental objects, the application and the session. The application object is common to all open sessions in an application on the same server. The user context is managed at the session level. A unique session identifier (StateID) is generated by the application server during the call for the first dynamic page. URL long or cookies will then systematically transmit this identifier each time a page is called. This enables the user context (variables, privileges, parameters, etc.) to be restored. The methods “put”, “get”, and “remove” make manipulating session object variables possible. Caution: if a dynamic page contains a link to a static page, the context is lost.

A dynamic page designates any HTML page containing server code. During compilation this page will generate a byte-code file (.hbb for HahtTalk, .class for Java) that the application server’s interpreter can run.

In looking at the server-side source code, we can see that the principle of dynamic pages is to reproduce HTML code in the HTTP flow by the intermediary of the instruction “print”.

When sending files (text, images, etc.) from the client station to an application server, it is necessary to use the File Upload mechanism. HahtSite offers a form field in its HTML components palette. In addition, an example that inserts an image in the database is provided and commented on in the help menu.

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

- **Editor**

Java or HahtTalk source editors offer syntax coloring. This allows for searches and cancellations, but not for completion, or property or method lists in a pop-up window.

Even if the information provided is often very good, assistance is not always contextual (only for some words).

It is also unfortunate that there is no mapping tool between the application’s business objects and interface, especially for a Java approach that turns towards object modeling.

- **Compilation and debugger**

Within the framework of an HahtTalk project, HahtSite uses the BasicScript compiler provided with the tool.

With Java, compilation is handled by the default compiler that is defined in the edit options. It is however possible to select a different compiler at the file-source level as long as the parameters for this compiler are positioned.

In either case, compiler returns error and alert information such as loss of a link, etc.

The debugger, for its part, provides vital services: stopping point, step by step, view of global and local variables as well as their modification. It also allows for procedure stepping.

It even works in a configuration with a remote (distributed) application server. All debugger activity is traced in a special log file on the application server and the number of debug sessions is limited.

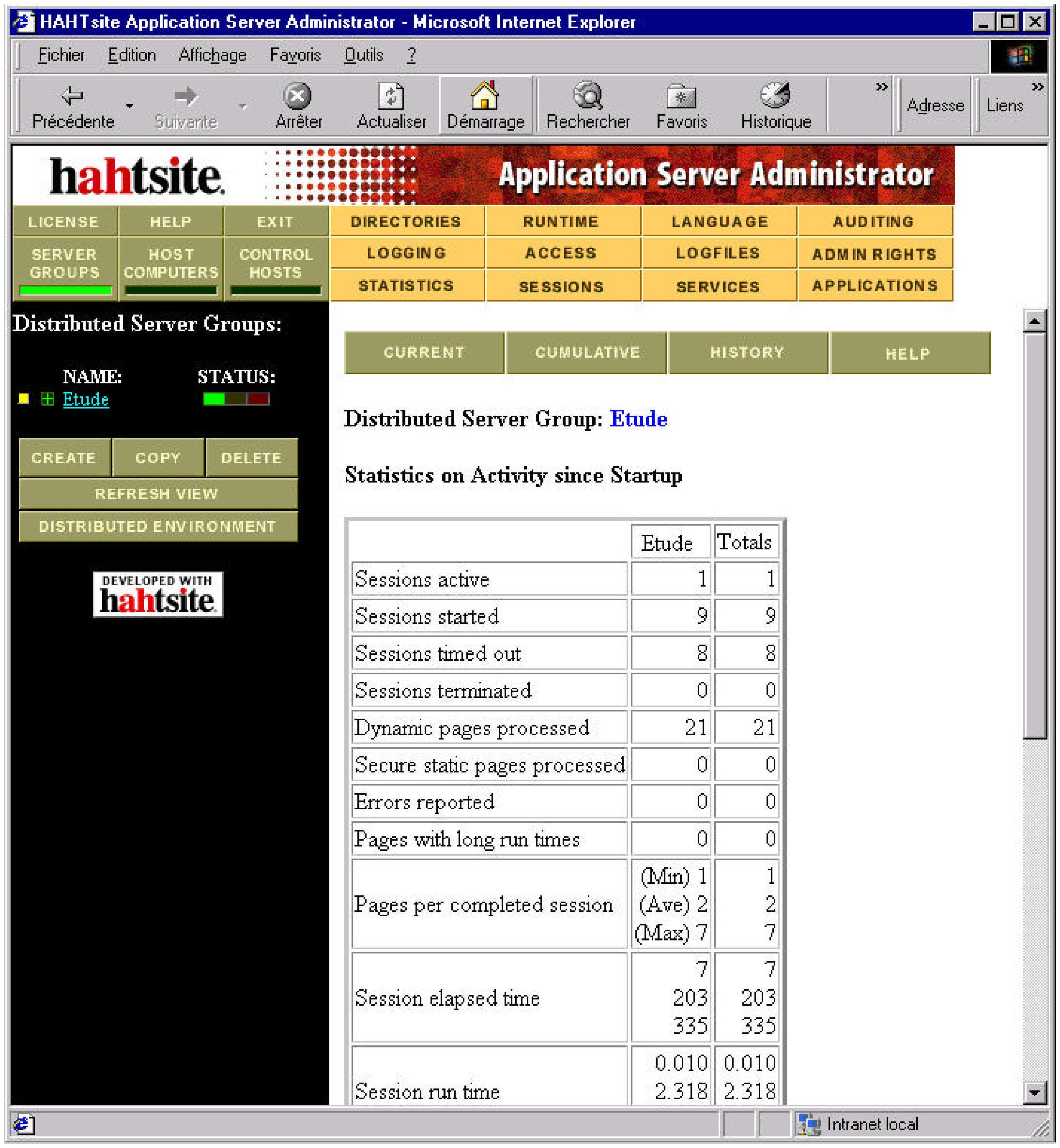
➢ Administration, optimization, and workload tests

The administration tool has the big advantage of having an HTML interface. It is thus possible to administer it from any station.

This tool makes it possible to define compiled file transfer repertories, the number of processes and threads, the JVM used, encryption implementation, etc., and of course, the stopping and restarting of instances.

The application server gives access to static information for each group of application servers:

- The current connections (number of sessions, pages, CPU time, etc.) per connected station with a total for all the stations
- The same information but from the application server’s launch
- History of a given period can be traced


HAHTsite Application Server Administrator - Microsoft Internet Explorer
Fichier Edition Affichage Favoris Outils ?
Précédente Suivante Arrêter Actualiser Démarrage Rechercher Favoris Historique Adresse Liens
hahtsite Application Server Administrator
LICENSE HELP EXIT DIRECTORIES RUNTIME LANGUAGE AUDITING
SERVER GROUPS HOST COMPUTERS CONTROL HOSTS LOGGING ACCESS LOGFILES ADMIN RIGHTS
STATISTICS SESSIONS SERVICES APPLICATIONS
Distributed Server Groups:
NAME: STATUS:
Etude
CREATE COPY DELETE
REFRESH VIEW
DISTRIBUTED ENVIRONMENT
DEVELOPED WITH hahtsite
CURRENT CUMULATIVE HISTORY HELP
Intranet local

Distributed Server Group: Etude

Statistics on Activity since Startup

	Etude	Totals
Sessions active	1	1
Sessions started	9	9
Sessions timed out	8	8
Sessions terminated	0	0
Dynamic pages processed	21	21
Secure static pages processed	0	0
Errors reported	0	0
Pages with long run times	0	0
Pages per completed session	(Min) 1 (Ave) 2 (Max) 7	1 2 7
Session elapsed time	7 203 335	7 203 335
Session run time	0.010 2.318	0.010 2.318

Application server administrator

It is also possible to implement an audit thus making it possible to trace, for a given user, the form field values, the Web server environment variables, dynamic page output, etc.

We slightly regret the lack of optimization solutions, a testing tool and a workload tool.

3.5.3. Evaluation

Server-side code generation assistance covers all of the tool's possibilities, whether we decide to code in Java or HahtTalk.

The simplicity and quality of the user context management is particularly noticeable, as is the creation of data source transactions.

The product has a few shortcomings when it comes to tools. Although the debugger carries out its task entirely, the editor could use some more work (completion, property and method pop-up windows, etc.)

The administration tool is simple and its HTML interface is quite practical. Optimization solutions, a testing tool and a workload tool would be welcomed additions.

IDE productivity for server processing	Rating/10
Server-side code generation assistance	7.5
Tools and automation provided by the IDE to improve productivity	5

3.6. Application server

3.6.1. Database access

HahtSite uses ADO 2.0 to access databases. Then, we have the possibility of using native drivers which can connect to MS-SQL Server 6.5 and 7.0, Oracle 7.x and 8.0, Informix 7.x and Sybase 10 and 11. If not, there is the ODBC solution which allows us to access, among others, MS-Access and micro databases. HahtSite also supports JDBC and OLE-DB.

SQL support is satisfactory as it allows for display/modification of standard blob fields (or long raw) and recognizes specific SQL orders to DBMSs (such as getdate() of MS-SQL Server).

The DataSource folder that is present in the project explorer groups together all database objects: tables, views, system tables, stored procedures and a new notion of shared queries. For all of these objects, a dialog box displays their properties:

- List of fields (with their type),
- General survey of all result values
- List of parameters (with their type) for stored procedures and shared queries

Only the possibility to display the source of a view or a stored procedure is not provided. Similarly, it is impossible to modify a table or a procedure. All of the features are natively accessible from the IDE or by using the ODBC.

Shared queries can be helpful when accessing a database that does not offer the possibility to make stored procedures because they have the disadvantage of not being able to be compiled on the DBMS. However, their use is very practical especially if we do not have a SQL query editor. A wizard makes it possible to build a query in a graphical manner, by proposing the list of fields, table joins, sorts, etc.

In order to use a database connection on a page, it is necessary to go through the DataAgent component. It acts as the link between the data source and the form fields that present this data. This component also allows for the definition of query field display formats.

There can be as many DataAgents as we would like on one page and each of them can be connected to DataSources referencing different databases.

In case of DBMS failure and restart, HahtSite is capable of restoring connections automatically as long as the method «getSharedConnectionByName » is used.

To monitor the number of connections that an application can create on a DBMS, HahtSite offers connection sharing. The definition of this parameter is linked to the number of processes defined on the application server; there cannot be more connections than processes. HahtSite also adds a configurable cache feature on the DBMS connection level, keeping a connection open during a defined period of time; this connection can then be reused, reducing the time necessary to open and to close the connection. This option can be useful for applications that open the connection at each page.

3.6.2. Language richness, openness

With this release of HahtSite, we have the choice between Java and HahtTalk (which is in fact VBA). Both of these languages necessitate an interpreter.

The choice between these two languages needs to be made at the beginning of the project. It depends more on the philosophy and culture of the company than on a difference in performance, because even if on paper Java appears to offer more possibilities and better performance, it is always possible to include Java classes in a project coded in HahtTalk.

With procedural aspects of third-generation languages and a strong object connotation, Java provides a large array of possibilities: abstraction, encapsulation, inheritance, polymorphism but also features and procedures. Most scalar-type data is supported. Tables are multidimensional and can be dynamic. Structures are supported.

A library of mathematical functions, string manipulation, including format conversion, etc. is present.

For its part, HahtTalk does not offer inheritance or polymorphism.

Variable management can be handled at session or application level with methods to add variables, modify them, delete them and read them. The application server defines time out by default, but it can be modified at the session object level. At the same time, a method of this same object makes it possible to finish the session.

➢ **Security**

SSL integration takes place within the repertories “subsites”: it suffices to define some repertories as “secure static” and to include the pages in them.

In an architecture with distributed application servers, HahtSite allows for easy integration of a firewall. A standard configuration makes it possible for the firewall to separate an “extranet” zone, in which “Foreground” application servers are found, from an “intranet” zone made up of “Control” and “Background” servers and which is the only one to keep sensitive information unique to the business and the user session.

Finally, the application server and the Web server communicate by using the encryption algorithm Triple-DES.

It is interesting to note the presence of page access profiles (see section 3.5 User Privileges) and an ESM destined to authenticate connections for e-commerce applications on Internet.

➢ **ERP (Enterprise Resources Planning)**

Openness to ERP is one of Haht Software’s key concerns. Through joint developments with editors, Haht Software offers solutions for connecting to the main ERP market players, integration middleware and transactional monitors, always using the same ESM principle:

- an ESM for SAP R/3 with BAPI and RFC support, jointly developed with SAP
- an ESM for DB2, CICS, VTAM, VSAM, IMS, provided by Neon Software
- an ESM for BEA Tuxedo

➢ **CORBA**

HahtSite's application server partially implements the CORBA 2.0 norm as it integrates Inprise's ORB VisiBroker for Java, filling the naming service and the event service.

The desire to install VisiBroker's must be specified at installation. It can thus act as a client and a CORBA server.

To be a CORBA client, it suffices to recover the object IDL and add it to the project.

A HahtSite application can be a CORBA server in two different ways:

- Traditional: the object instances provide a service than can be called from different clients. In this example, the application is launched as a service at the application server level.
- Session-oriented: a group of distinct instances is created for each application session.

➢ **But also**

The application server makes it possible to be a DCOM client and to instantiate ActiveX components thanks to the "CreateObject" method.

ESM allows for the integration of LDAP, making the implementation of "single sign-on", work groups, etc. possible.

3.6.3. Deployment

➢ **Multiple platforms**

The application can be deployed on an application server installed on Windows NT 4.0, Solaris Sparc 2.5.1, HP-UX 10.20 or AIX 4.2.1. It is nonetheless important to know the deployment target beforehand to avoid using APIs specific to an operating system (this information is indicated in the help menu of each feature).

It is unfortunate that the application server is not available on Linux and that nothing has been announced to let us think that this might soon be the case. Digital Unix is not supported either.

The idea of site greatly facilitates deployment. Publication sites are accessible from the IDE project explorer. The publication consists only in copying static components (pages, style sheets, etc.) on the Web server's "DocumentRoot" and the compiled processing (dynamic pages, code, etc.) on the application server's project repertory.

The application server is not physically linked to the HTTP server. It can be made up of several application servers spread across several machines, but belonging to the same group of servers. It supports CGI 1.1 but also NSAPI 2.0 and 3.0 and ISAPI, making it possible to communicate with the principle Web servers on the market (Apache, Netscape FastTrack and Enterprise Server, Microsoft Information Server, etc.).

➢ **Failover**

Failover and restart support depends on the selected configuration. To increase application isolation, it is necessary to create several server groups on the same physical server.

Restart in case of failover is handled by replication: the “control host” can be associated with a backup control host that will take over. A minimum of two “foreground” hosts on different machines and especially “background “ hosts are necessary as they guarantee a transfer of user sessions and thus guarantee that the context will be saved.

➢ **Load balancing**

In a distributed configuration, load balancing is based on the modularity of three types of application servers (“control”, “foreground” and “background”).

The control host communicates with foreground and background hosts to balance the load of each in considering their respective weights. It is the only parameter allowing for load balancing configuration. The rest is left to the application servers that take a decision in relation to the information gathered concerning the load of each server.

3.6.4. Evaluation

The data access module makes it possible to cover the whole array of possibilities offered by the main DBMSs on the market.

With support for Java as a new server-side programming language, HahtSite opens itself up to new horizons and brings a touch of object method to its IDE.

The main axes toward which HahtSite turns are e-business (security), ERP and CORBA.

Deployment facility has been one of HahtSite’s strengths for a long time. This new release of the tool confirms this, with the contribution of an even more modular architecture, even if some regret the fact that the application server is not available on Linux.

Application server	Rating/10
Database access	**8.8**
Richness of language, openness	**6.9**
Deployment	**7.9**

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 intranet architecture chosen

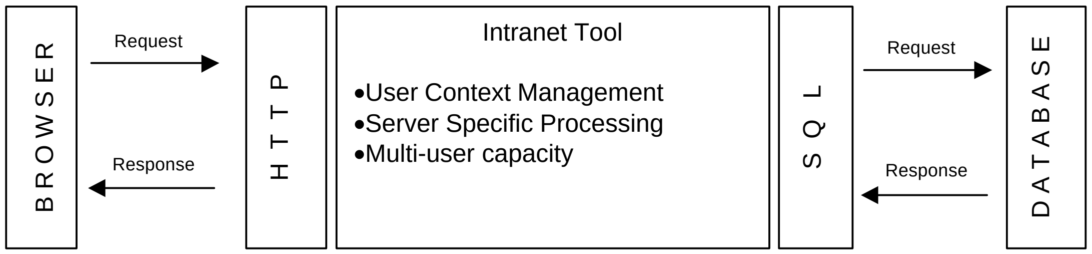
The application's clients are standard browsers (the application generates only HTML).

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.

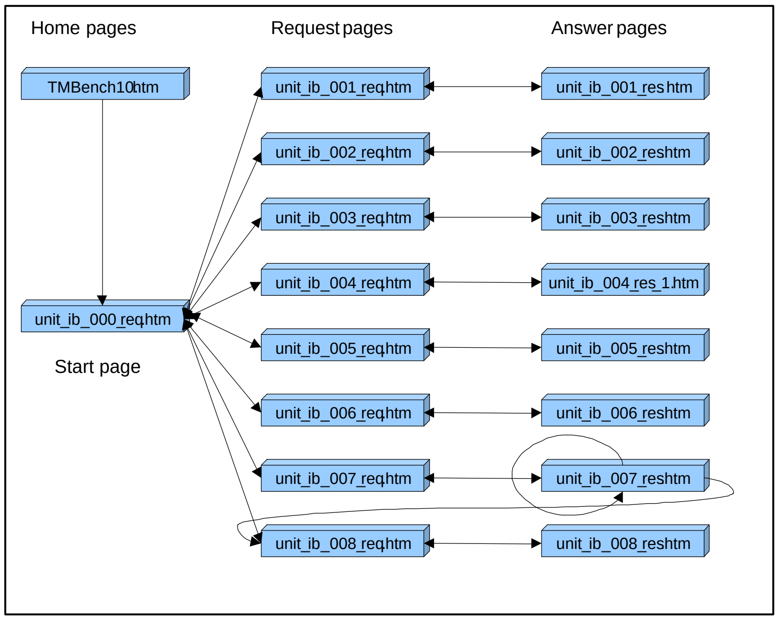
➢ Database and SQL queries

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

Technical constraints

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

- HTML files 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 link (previous page) enabling 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:

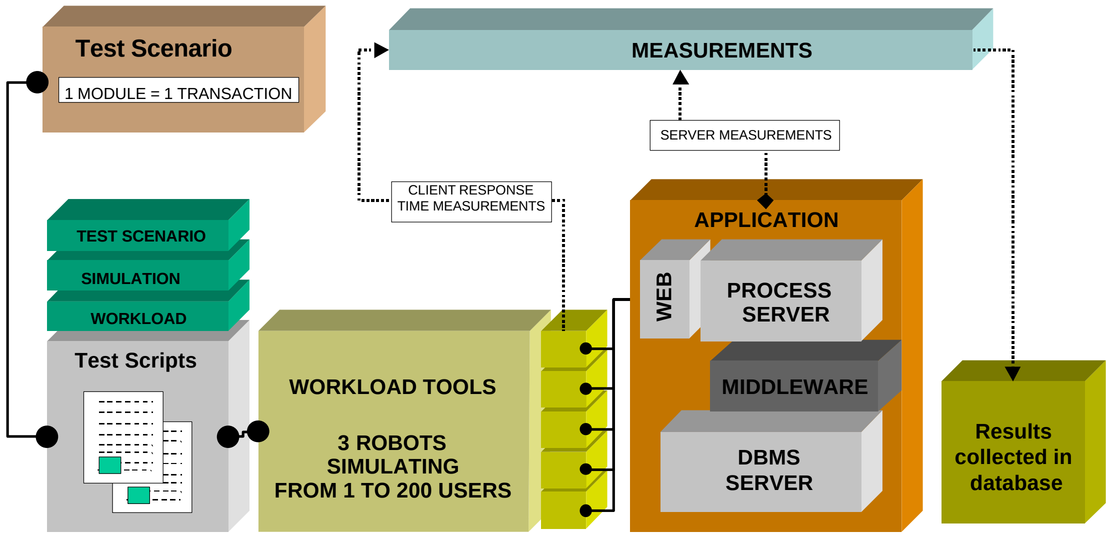
- Resulting data are to be put in tables as soon as possible.
- The default table border must be equal to one.
- No specific format is to be applied to displayed data.

➢ 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 the 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 up, which expresses the load generated by the application server.
- On the DBMS server, by counting the number of connections opened by the application, as well as 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. Development environment productivity

➢ **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).	☆☆	All of the editors (HTML, VB or Java server code, client code, images, image maps, and style sheets) are accessible from the same user interface.
2	Printed documentation and online help with search options.	☆☆	Printed documentation includes the following volumes: - for the IDE: an installation manual, a programming manual, a user's manual and a Widget programming manual, - for the application server: an installation manual and an administration manual Online administration is accessible from the menu bar (or with the F1 key) and includes an index and a key-word search.
3	Ease of use and installation. The installation mustn't call for too many prerequisites (browser, DBMS, HTTP server). The developer must be able to use the wizards quickly.	☆☆	An «InstallShield » installs the IDE easily: it starts by updating the JVM, if necessary by installing Microsoft's JVM 5.00.3165. If you wish to develop with Java, it is recommended to install a JDK beforehand, as the IDE will detect it and integrate it into its parameters automatically. Complete installation begins by installing Apache 1.3.4; then the IDE by selecting the default browser (Netscape or IE); finally comes the installation of a an application server dedicated to testing with automatic detection of pre-installed HTTP servers and publication site configuration. Installation of the distributed application server is just as easy on NT. Deinstallation poses no problem; one must not forget to delete the services in the database registries. However, application server installation on Unix (Aix 4.3.2) is not as easy and requires a numerous Unix-specific prerequisites (file system, group, user).
4	Richness and quality of the tutorials and examples provided. The examples must cover all internet-related problems (script language, DBMS access, context management, etc.).	☆	A «Getting started » menu offers a "Shockwave" presentation and a complete lesson in HTML format: it proposes 11 lessons which become more and more difficult as well as two applications (VB and Java) that group together a large part of the tool's features.

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.	☆☆	In terms of development, the Haht Software product provides an IDE (Integrated Development Environment) that allows for the generation of server-side HahtTalk (VB) code and/or Java and client code (JavaScript, VB Script). There is also an HTML editor in WYSIWYG mode (tag mode) and an image editor. A development application server is provided, as is the Apache 1.3.4 HTTP server. Finally, a multiple-user application server is necessary for deployment and test site.
2	Presence of a project and project resource manager, quality of the interface with PVCS, VSS, etc, (check-in/check-out, versioning).	☆	HahtSite does not provide developers with profile features, work group tools or versioning tools. It allows, among other things, for interfacing, with a good level of integration, with Microsoft's Visual Source Safe, but also with Merant Intersolv PVCS, StarBase StarTeam and MKS Source Integrity.
3	Tool for developing multilingual applications (listed in a dictionary, post-development extraction).	☆	A multilingual solution is possible with ESMs (Enterprise Solution Modules are components that can be downloaded from the Haht Software Web site) NLS 3.0 (National Language Support). The translation depends on browser configuration; one can also impose a language using "functions".
4	Test HTTP server and a local development DBMS.	☆	Development distribution is provided with Apache 1.3.4. However, it does not include a DBMS.
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 controlled breaks and offer a test mode.	☆	It is possible to make reports in HTML by using either a page wizard or by going through the Widgets "ADO Report Maker" and "ADO Report Next/Previous Link". An ESM (ESM for Seagate Crystal Reports) allows for easy report integration. It is not necessary to install Crystal Reports 5.0 (or 6.0) on all development stations, but only on those that will edit the reports (.RPT). An ESM for Adobe Acrobat makes it possible to generate PDF format dynamic reports (that can navigate throughout all data) and to modify them in Adobe Acrobat Exchange.
6	Presence of a search engine and/or quality of the interface with a search engine.	💣	No search engine. The only solution is to configure that of the HTTP server.
7	Presence of a relational and/or object modeling tool, possibility of interfacing with Power AMC, Rational Rose, Mega, etc.	💣	HahtSite does not have any data modeling tools and does not offer an interface with another market tool; it only offers a view of the database set-up (tables, views, stored procedures, shared queries, but no table index) without being able to play with the definition of these objects.
8	SQL query editor, with support for stored procedures (run and display source code). The editor must allow for query entry and provide, in real time, without passing by the application server, the result of this query.	☆	The only way to query the database is to create a shared query. This notion allows for the generation of SQL code that can be reused by the editor with the possibility of graphical assistance (list of tables, fields, joins, etc.)
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.	☆☆	The project explorer offers a general overview of the project with all of the elements that make it up. It is possible to create repertories that group together components such as HTML pages, server code files, client script files, style sheets, images, and also Widgets, database connections and publication sites. A "Design" view gives a graphic view of how pages are linked together. It also makes HTML page creation using templates easier, with drag&drop from the palette. Finally, it can give an order to pages within a repertory. This makes their linking easier in the case of a static site.

10	Repository, cross object references (graphics, components, objects used by the IDE, etc.), 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.	💣	Relationships between components (static HTML pages, images, Java classes, Widgets, style sheets) are presented graphically by the action "Show References" (don't forget an "Update Dependencies" to update the information). HahtSite does not possess an exhaustive dictionary that references all project components; such as calls between pages are only made in a dynamic manner (to preserve the context); "Show References" does not allow us to establish complete application kinematics. HahtSite does not offer any information on unused declared variables either (not even on the compiler level); the only solution is to use "find in files" ("grep" on project files). Deleting a "database" object does not inform the developer of its use.

➢ 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.	☆☆	The HTML page editor is WYSIWYG; one chooses the form fields from the graphics palette. The project components (images, HTML pages, Widgets) are inserted in the page by drag & drop from the project explorer. This editor makes it possible to define style sheets (CSS1), clips (reusable HTML code) and frames. The image editor recognizes GIF, JPEG, and PNG formats. An image map editor can reference project components, even though it is not integrated in the IDE.
2	Numerous table manipulation possibilities (cell mergers, cell breakdown, nested tables)	☆☆	The tool makes it possible to graphically merge cells horizontally and vertically, split cells and to include a table within a cell of another table.
3	Presence of a repository for graphics components (objects or groups of objects).	☆☆	Besides the standardly provided form fields, it is possible to define "Clips" or reusable parts of HTML pages (for example, headers and footers) that can be used anywhere on a page and can be nested.
4	Possibility of creating style sheets (not necessarily CSS generation, simply on the IDE level), templates, and page models.	☆☆	It is possible to define style sheets in respecting the CSS1 (Cascading Style Sheet Level 1) standard and in partially respecting CSS2. Creating templates (that will add themselves to the other page templates) from an HTML page is possible (in the IDE options, it is necessary to configure the template repertory as being shared by all developers).
5	HTML code generation during the insertion of Java or ActiveX components and images in HTML pages.	☆☆	Inserting a Java applet in an HTML page is possible from the "Insert Object" menu; applet properties can be indicated (position, parameters, style, etc.). To insert an ActiveX component in an HTML page, the use of the "ActiveX" Widget is necessary: one can thus select it in a list of all the ActiveX components present on the machine. A dialogue makes it possible to configure the component. The tool generates the adapted HTML code.
6	Client-side generation of monitoring scripts.	☆☆	Client-side monitoring scripts (JavaScript, VBScript) are generated automatically for the form fields; they allow for the validation of entry zones (text, numeric, date, and mask).

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.	☆☆	The HTML editor is WYSIWYG and allows for page display; it can be configured to choose object visibility (code, images, objects, etc.). We can also display a page in a browser among those referenced in the configuration options. The HTML source editor provides the same automatic indent (when one passes in WYSIWYG mode and then goes back to tag mode), find/replace functions, syntax correction and coloring.
2	Presence of dynamic HTML page generation wizards.	☆☆	There are several page creation wizards (connection, master/detail, form, and list); they are configurable and can be modified after their generation.
3	Import/export of HTML pages and total command of HTML (static and dynamic, in other words generating HTML code by programming).	☆☆	Importing makes it possible to copy an HTML page or even a repertory or site. A project property allows for the definition of source visibility levels (none, possibility to make links with URL, possibility to share code with another project – same site, same session). Export passes through page save as a file. For all pages (static, dynamic, wizard generated), one keeps control over HTML code. In the description of each graphic object there is a "custom" tab that lists all of the HTML tags for this object and the properties attributed to it. HahtSite also offers the possibility of modifying HTML code generation. One thus has total control over the generated HTML.
4	Assistance in entering HTML (tag list, completion, documentation, etc.) and client-side script language (events, objects, authorized attributes and methods, documentation, etc.)	☆	The HTML editor offers neither a tag list nor completion: only script monitor confirms the entered code's validity. For client script code, HahtSite provides entry help, by presenting the DOM (Document Object Model) of the page and all the events offered in relation to the browsers.
5	Presence of client-side script language components and Java applets.	☆☆	There are many JavaScript files, which are used for form field validation checks. These can be personalized to display messages in a different language. HahtSite also provides some Java applets.
6	HTML 4.0, DHTML and DOM are taken into account.	☆☆	HahtSite supports HTML 4.0, DHTML (DIV, SPAN, etc.) and DOM entirely. A DHTML example (Layer and SPAN) is provided. HahtSite offers client script entry assistance by recovering the page's DOM (Distributed Object Model) in the form of a hierarchical tree; here we find all of the possible event pages and the present fields (with an indication concerning compatibility with the main browsers and their versions) and manually coded routines.

➢ Java graphical interface generation

Step 5: Java graphical interface elaboration automation and assistance

N°	Description	Rating	Comments
1	Presence of a Java WYSIWYG editor.	💣	No.
2	Presence of a components palette (AWT, JFC, etc.)	💣	No.
3	Possibility of creating JavaBean components.	💣	No.
4	Possibility of creating Java applets.	💣	Yes, but no wizard nor JDK is provided.
5	Possibility of enhancing the Java graphics object palette and the presence of an objects inspector.	💣	No.
6	Presence of dynamic transactional application generation wizards with a Java graphical interface.	💣	No.

➢ IDE Productivity for server processing

Step 6: Server-side code generation assistance

N°	Description	Rating	Comments
1	Management of user profiles with application and page access security.	☆☆	To manage user profiles, it suffices to define “Privileges” on the project level and to attribute them to pages. The code that monitors page access is thus generated automatically. The only thing left to do is to program using the “addPrivilege” and “removePrivilege” commands on the “session” object level in order to attribute rights.
2	Configurable transactional application creation wizards with the possibility of touching up the code after it has been generated.	☆☆	Wizards make it possible to create pages (forms, lists, etc.) which handle data source transactions. To personalize these wizards, it is necessary to create one’s own components (Widgets).
3	Assistance in creating standard distributed objects (ActiveX, CORBA, and EJB).	💣	Creating ActiveX components or EJBs is not possible. A CORBA object IDL can be defined, but HahtSite offers no assistance whatsoever.
4	Server-side assistance in sending e-mail (high-level API, examples, documentation).	☆☆	The “Smtp” and “Message” objects make it possible to send an e-mail from the application server by specifying a minimum amount of information (gateway, addressee, sender, subject, contents). It is also possible to attach a document. An example is provided within the help section.
5	Automatic management of the unique session identifier.	☆☆	The identifier (stateID) is generated by the application server when one makes reference to a dynamic page for the first time. It then allows the user context to be restored.
6	Freedom to choose the context management used for transfer of the identifier (URL longs, cookies, and hidden variables).	☆☆	HahtSite offers the choice between URL long and cookies for context management; this parameter is one of the properties of publication site definition.

7	Freedom to choose between a client-side and server-side stored context, with the possibility of tracing the history.	💣	No. Context management is only handled server-side.
8	Assistance in File Uploading (interface, examples of code, explanations).	☆☆	A form field makes it possible to define the HTML interface. An example is given in the help.

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

N°	Description	Rating	Comments
1	Good quality source editor, with find and replace features, context help, completion, syntax parser, etc.	☆	The source editor offers syntax coloring. Contextual aid and completion would make this editor impeccable.
2	Complete debugger (logs, step by step, choice of the application or tracing in functions/methods/procedures, modification of variable content).	☆☆	The debugger makes it possible to progress step by step, and to know the content of local and global variables. It also allows for modification of variable content. It leaves one the choice of tracing the already browsed features or not. It should be noted that the application server includes a log and a log for debugs mode.
3	Assistance in mapping between business objects and the application interface.	💣	There are not any tools that allow mapping between Java classes (could act as business objects) and the application interface.
4	Presence of an application administration tool.	☆☆	Application servers can be managed from a distance thanks to their administration tool's HTML interface. These parameters can be modified on the fly.
5	Presence of an application use/consumption administration tool with graphics and optimization solutions.	☆	The application server gives access to statistics for each group of application servers: - current connections (number of sessions, number of pages, CPU time, etc.) per connected station, with a total for all stations - the same information but from the launch of the application server, - history of a given period can be traced. It is also possible to implement an audit, thus tracing form field values, Web server environment values, dynamic page output, etc. for a given user. Neither graphs nor proposed solutions for reducing response time.
6	Presence of a testing tool and/or workload tool.	💣	No.

4.2.2. Application server productivity

Step 8: Database access

N°	Description	Rating	Comments
1	Support of ODBC and/or JDBC type 1 driver.	☆☆	HahtSite uses ADO 2.0 to access databases by ODBC. It is an ADO-specific implementation that is used whether the project is developed with HahtTalk or Java.
2	Native database access (SQL Net, Dblib-CTLib, Inet) and/or via type 2 and/or type 4 JDBC.	☆☆	HahtSite uses ADO 2.0 to access the following databases in native: MS-SQLServer 6.5 and 7.0, Oracle 7.x and 8.0, Informix 7.x and Sybase 10 and 11.
3	Support of Blobs/Long Raw (reading and insertion).	☆☆	It is possible to display, modify and insert standard Blobs/Long Raw fields. However, this can not be done using "DataAgents", it is necessary to establish a connection with the database and use a RecordSet.
4	Support of DBMS-specific SQL queries (Ex: select getdate (), select banner from v$version, etc).	☆☆	The queries «select getdate() » for Sybase and «select banner from v$version » for Oracle work.
5	Support of stored procedures, with the transfer of in/out parameters.	☆	Stored procedures are grouped together on an object connection that identifies a connection to a data source with a diagram; the object connection also contains tables, views and shared queries. It is necessary to update the diagram in order to bring the list of object connections up to date; this operation even specifies changes made to an object.
6	Support of 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.	☆☆	Connection restart takes place as long as the method "getSharedConnectionByName" is used. This is the method automatically used by the wizards.
7	Support of multiple data sources within one page or graphical window. Possibility of creating standalone connections.	☆☆	Access to a data source from a page is handled by the intermediary of DataAgents, there can be as many DataAgents as one would like on a page and each one can be connected on a different connection. One can also connect in a controlled manner, by establishing a database connection within the code.
8	Extended management of connection pools (multi-instances and maximum number of connections).	☆	Connection pool management in HahtSite ("Connection sharing") is associated with a number of processes configured for the application server ("server group"); that thus defines the maximum number of connections that the sessions must share. No multi-instance forecasting connections are possible. Connection caching backs up the idea of database access optimization; which consists in indicating the period of time during which a database connection is kept to be possibly reused by another query (useful for applications that open and close their connection with each page).

Step 9: Language richness, openness

N°	Description	Rating	Comments
1	Creation of functional classes that support object polymorphism.	☆☆	Java supports polymorphism and Java classes can be included in an HahtTalk project.
2	Support of all variable types (dates, multi-dimensional tables, dynamic tables, and structures).	☆☆	With HahtTalk, the flaw concerns defining structure tables. No problem in Java.
3	Creation of features, with transfer of variables and recursive elements.	☆☆	Possibility of creating features and procedures in Java and HahtTalk, they can even be recursive.
4	Variable management at session and application level.	☆☆	Variable management can be handled at session or application level using the “put” method.
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).	☆☆	Yes.
6	Presence of complete log-out 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 force log out, and to launch server-side processing if the session expires.
7	Server-side generation of print files using values coming from the data source.	☆	Generation of print files goes through a specific development; indeed nothing is planned for this. However, it is possible to dynamically feed PDF files.
8	Server-side generation of image files (GIF, standard formats) using values coming from the data source.	💣	No tool for generating GIF images; the solutions can turn to a component such as ChartFX (which thus only works via OLE on NT) or other solutions such as “shared library” on Unix.
9	Capacity to handle File Upload (the tool must allow for the recovery of HTTP flow in a string for example).	☆☆	Yes.
10	Presence of an object/relational mapping module.	💣	No notion of business object mapping on the database.
11	Presence of a distributed objects system.	☆☆	The application server implements the CORBA 2.0 standard as it integrates Inprise’s VisiBroker for Java, playing the role of an ORB, Naming Service and Event Service (it is necessary to specify one’s desire to install VisiBroker at the time of installation). In this way, it can behave as a client and a CORBA server.
12	Possibility of interfacing with a standard distributed object model (DCOM, CORBA, and EJB).	☆☆	Beyond CORBA support by VisiBroker, HahtSite’s application server also makes it possible to be DCOM client and instantiate ActiveX components thanks to the “CreateObject” method.
13	SMTP and POP3/IMAP4 and LDAP support.	☆	SMTP and POP3 are supported; it suffices to use the objects (“Message” and “Pop”). However, IMAP4 is not implemented. For LDAP it is necessary to go through an ESM.
14	The processing server is an HTTP and FTP client.	☆	Low-level API.
15	Possibility of proposing different levels of security for applications (SSL, SET, use of firewalls, etc.).	☆☆	SSL integration takes place on the “subsites” repertory; it suffices to place the pages concerned in this repertory. Some pages can not be directly accessed by a bookmark; a parameter must be specified in the page properties. In a distributed application server architecture, HahtSite allows for the easy integration of a firewall: a standard configuration allows the firewall to separate a zone “extranet” in which one finds “Foreground” application servers from a zone “intranet” made up of “Control” and “Foreground” servers and which is the only one to keep information that is sensitive to business and user session. Finally, the application server and the Web server communicate using the encryption algorithm, Triple-DES.

16	Openness to ERP, with «business » modules adapted to ERP (SAP, Peoplesoft, etc.).	☆	It is possible to download an ESM (Enterprise Solution Module) from Haht Software's Web site. This ESM makes it possible to easily connect to SAP R/3; this module provides, among others, components and wizards that support BAPI/RFC. HahtSite sells two e-Scenario applications created in cooperation with SAP.
17	Openness to transactional monitors (Tuxedo, Encina, etc.), integration middleware (MQ Series, etc.).	☆	Another ESM "Shadow Direct" developed in collaboration with Neon Systems makes it possible to interface with such mainframes as CICS, IMS, DB2, VSAM, etc. It is also possible to interface with Tuxedo via an ESM.
18	Consideration of XML (XML application server, reuse of XML components, etc.).	💣	An XML parser is available with HahtSite. Unfortunately, the documentation is only available in "white paper" format for the moment.

Step 10: Deployment

N°	Description	Rating	Comments
1	Multi-platform application server: Unix, Linux, and Windows NT.	☆	For the application server, there is the choice between Windows NT 4.0, Solaris Sparc 2.5.1, HP-UX 10.20, and AIX 4.2.1. No Linux version planned. The notion of site makes deployment much easier. However, environment-specific configuration (DSN ODBC, alias of Web servers, etc.) is not taken into account.
2	Support of several types of interface with HTTP servers: CGI, ISAPI, and NSAPI.	☆☆	The application server supports CGI 1.1 and is totally independent of the HTTP server which can be either of the following: Apache, Commerce builder, Microsoft + ISAPI, NCSA, Netscape + NSAPI 2.0 and 3.0, Oracle Web Request Broker, Purveyor, Spry, Website, WebSTAR.
3	Possibility of deployment on other application/object servers (ASP, LiveWire, NCA cartridge, EJB, JSP, servlets etc.).	💣	Impossible to deploy the application on other servers on the market.
4	Support of application-level and session-level failover.	☆☆	Failover and restart support depends on the selected configuration: - application isolation can be improved by the creation of several groups of servers, - failover is handled by replication: the "control host" can be associated with a backup control host; the multiplicity of "foreground" and especially "background" hosts guarantees user session transfer and thus that the context will be saved.
5	Possibility of modifying the application without stopping the services («on the fly » modifications).	☆☆	An application can be updated at any time without having to stop the slightest service. Just be careful to empty the browser caches to have the updates.
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.	☆☆	The application server is physically linked to neither the HTTP server nor data server. It can be made up of several application servers spread out on several machines, but belonging to the same group of servers.
7	Support of load balancing, with the possibility of configuring the balancing algorithm.	☆☆	The principle of load balancing is based on the modularity of application servers. Three kinds of application servers exist ("control", "foreground" and "background") and can be answered on different systems. The control host communicates with the foreground and background hosts to balance the load of each of them, taking their respective balancing into account. This is the only parameter allowing for configuration of load balancing. The rest is handled by the application servers, which make a decision in relation to the information received.

A study conducted by

TechMetrix Research

Authors
Franck Gonzales
Jean-Yves Haguet

Translation and editing
Gina Faucher

Contributors
Emmanuel Gourion
Christophe Lauer
Philippe Mougin

Publication date: September 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.fr
r&d@sqli.fr

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